# Agilent Technologies
# E1468A/E1469A
# Relay Matrix Switch Modules
# User's Manual

Agilent Technologies

Manual Part Number: E1468-90005
Printed September 2012
Printed in Malaysia E0912

# Contents

## E1468A/E1469A Relay Matrix Switch User's Manual


**Front Matter** ..... **7**
- Agilent Technologies Warranty Statement ..... 7
- U.S. Government Restricted Rights ..... 7
- Safety Symbols ..... 8
- Warnings ..... 8
- Documentation History ..... 8
- Declaration Of Conformity ..... 9

**Chapter 1 - Getting Started** ..... **11**
- Using This Chapter ..... 11
- Relay Matrix Switches Description ..... 11
  - E1468A Switch Description ..... 11
  - E1469A Switch Description ..... 11
  - E1468A/E1469A Connector Pin-Outs ..... 11
- Configuring the Relay Matrix Switches ..... 15
  - Warnings and Cautions ..... 15
  - Setting the Logical Address Switch ..... 16
  - Setting the Status Register Switch ..... 16
  - Setting the Interrupt Priority ..... 17
  - Installing Relay Matrix Switches in a Mainframe ..... 18
- Configuring the Terminal Modules ..... 20
  - Wiring the Terminal Modules ..... 20
  - Creating Larger Matrixes ..... 23
  - Attaching a Terminal Module to the Relay Switch Module ..... 27
- Programming the Relay Matrix Switches ..... 28
  - Using SCPI Commands ..... 28
  - Addressing the Modules ..... 28
  - Initial Operation ..... 29

**Chapter 2 - Using the Relay Matrix Switches** ..... **31**
- Using This Chapter ..... 31
- Relay Matrix Switch Commands/States ..... 31
  - Relay Matrix Switch Commands ..... 31
  - Relay Matrix Switch Query Commands ..... 32
  - Power-on and Reset Conditions ..... 32
- Relay Matrix Switch Functions ..... 33
  - Checking Module Identification ..... 33
  - Switching Channels ..... 33
  - Recalling and Saving States ..... 34
  - Detecting Error Conditions ..... 35
  - Synchronizing Relay Matrix Switches ..... 36

**Chapter 3 - Relay Matrix Switch Command Reference ..... 37**
About This Chapter ..... 37
Command Types ..... 37
Common Command Format ..... 37
SCPI Command Format ..... 37
Linking Commands ..... 39
SCPI Commands Reference ..... 39
ABORt ..... 40
ARM ..... 41
ARM:COUNt ..... 41
ARM:COUNt? ..... 42
INITiate ..... 43
INITiate:CONTinuous ..... 43
INITiate:CONTinuous? ..... 44
INITiate[:IMMediate] ..... 44
OUTPut ..... 45
OUTPut:ECLTrg[:STATe] ..... 45
OUTPut:ECLTrg[:STATe]? ..... 46
OUTPut[:EXTernal][:STATe] ..... 46
OUTPut[:EXTernal][:STATe]? ..... 47
OUTPut:TTLTrg[:STATe] ..... 48
OUTPut:TTLTrg[:STATe]? ..... 49
[ROUTe:] ..... 50
[ROUTe:]CLOSe ..... 50
[ROUTe:]CLOSe? ..... 51
[ROUTe:]OPEN ..... 52
[ROUTe:]OPEN? ..... 53
[ROUTe:]SCAN ..... 53
STATus ..... 55
STATus:OPERation:CONDition? ..... 56
STATus:OPERation:ENABle ..... 57
STATus:OPERation:ENABle? ..... 57
STATus:OPERation[:EVENt]? ..... 58
STATus:PRESet ..... 58
SYSTem ..... 59
SYSTem:CDEScription? ..... 59
SYSTem:CPON ..... 59
SYSTem:CTYPe? ..... 60
SYSTem:ERRor? ..... 60
TRIGger ..... 62
TRIGger[:IMMediate] ..... 62
TRIGger:SOURce ..... 63
TRIGger:SOURce? ..... 64
IEEE 488.2 Common Commands Quick Reference ..... 65
SCPI Commands Quick Reference ..... 66

**Appendix A - Relay Matrix Switch Specifications ..... 67**

**Appendix B - Register-Based Programming ..... 69**
About This Appendix ..... 69
Register Addressing ..... 69
Addressing Overview ..... 69
The Base Address ..... 70
Register Definitions ..... 72
Reading the Registers ..... 72
Manufacturer Identification Register ..... 72
Device Identification Register ..... 73
Status/Control Register ..... 73
Relay Control Registers ..... 73
Writing to the Registers ..... 73
Status/Control Register ..... 73
Relay Control Registers ..... 74

**Appendix C - Relay Matrix Switch Error Messages ..... 77**

**Appendix D - Relay Life ..... 79**
Replacement Strategy ..... 79
Relay Life Factors ..... 79
End-of-Life Determination ..... 79

**Index ..... 81**

## AGILENT TECHNOLOGIES WARRANTY STATEMENT

**AGILENT PRODUCT:** E1468A/E1469A Relay Matrix Switch Modules **DURATION OF WARRANTY:** 3 years

1. Agilent Technologies warrants Agilent hardware, accessories and supplies against defects in materials and workmanship for the period specified above. If Agilent receives notice of such defects during the warranty period, Agilent will, at its option, either repair or replace products which prove to be defective. Replacement products may be either new or like-new.

2. Agilent warrants that Agilent software will not fail to execute its programming instructions, for the period specified above, due to defects in material and workmanship when properly installed and used. If Agilent receives notice of such defects during the warranty period, Agilent will replace software media which does not execute its programming instructions due to such defects.

3. Agilent does not warrant that the operation of Agilent products will be interrupted or error free. If Agilent is unable, within a reasonable time, to repair or replace any product to a condition as warranted, customer will be entitled to a refund of the purchase price upon prompt return of the product.

4. Agilent products may contain remanufactured parts equivalent to new in performance or may have been subject to incidental use.

5. The warranty period begins on the date of delivery or on the date of installation if installed by Agilent. If customer schedules or delays Agilent installation more than 30 days after delivery, warranty begins on the 31st day from delivery.

6. Warranty does not apply to defects resulting from (a) improper or inadequate maintenance or calibration, (b) software, interfacing, parts or supplies not supplied by Agilent, (c) unauthorized modification or misuse, (d) operation outside of the published environmental specifications for the product, or (e) improper site preparation or maintenance.

7. TO THE EXTENT ALLOWED BY LOCAL LAW, THE ABOVE WARRANTIES ARE EXCLUSIVE AND NO OTHER WARRANTY OR CONDITION, WHETHER WRITTEN OR ORAL, IS EXPRESSED OR IMPLIED AND AGILENT SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTY OR CONDITIONS OF MERCHANTABILITY, SATISFACTORY QUALITY, AND FITNESS FOR A PARTICULAR PURPOSE.

8. Agilent will be liable for damage to tangible property per incident up to the greater of $300,000 or the actual amount paid for the product that is the subject of the claim, and for damages for bodily injury or death, to the extent that all such damages are determined by a court of competent jurisdiction to have been directly caused by a defective Agilent product.

9. TO THE EXTENT ALLOWED BY LOCAL LAW, THE REMEDIES IN THIS WARRANTY STATEMENT ARE CUSTOMER'S SOLE AND EXLUSIVE REMEDIES. EXCEPT AS INDICATED ABOVE, IN NO EVENT WILL AGILENT OR ITS SUPPLIERS BE LIABLE FOR LOSS OF DATA OR FOR DIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL (INCLUDING LOST PROFIT OR DATA), OR OTHER DAMAGE, WHETHER BASED IN CONTRACT, TORT, OR OTHERWISE.

FOR CONSUMER TRANSACTIONS IN AUSTRALIA AND NEW ZEALAND: THE WARRANTY TERMS CONTAINED IN THIS STATEMENT, EXCEPT TO THE EXTENT LAWFULLY PERMITTED, DO NOT EXCLUDE, RESTRICT OR MODIFY AND ARE IN ADDITION TO THE MANDATORY STATUTORY RIGHTS APPLICABLE TO THE SALE OF THIS PRODUCT TO YOU.

## U.S. Government Restricted Rights

The Software and Documentation have been developed entirely at private expense. They are delivered and licensed as "commercial computer software" as defined in DFARS 252.227- 7013 (Oct 1988), DFARS 252.211-7015 (May 1991) or DFARS 252.227-7014 (Jun 1995), as a "commercial item" as defined in FAR 2.101(a), or as "Restricted computer software" as defined in FAR 52.227-19 (Jun 1987)(or any equivalent agency regulation or contract clause), whichever is applicable. You have only those rights provided for such Software and Documentation by the applicable FAR or DFARS clause or the Agilent standard software agreement for the product involved.

**Agilent Technologies**

E1468A/E1469A Relay Matrix Switch Modules User's Manual
Edition 5

## Documentation History

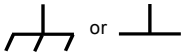
All Editions and Updates of this manual and their creation date are listed below. The first Edition of the manual is Edition 1. The Edition number increments by 1 whenever the manual is revised. Updates, which are issued between Editions, contain replacement pages to correct or add additional information to the current Edition of the manual. Whenever a new Edition is created, it will contain all of the Update information for the previous Edition. Each new Edition or Update also includes a revised copy of this documentation history page.

Edition 1 . . . . . November, 1990
Edition 2 . . . . . April, 1993
Edition 3 . . . . . November, 1994
Edition 4 . . . . . February, 1996
Edition 5 . . . . . December, 2000
Edition 5 Rev. 1 . . . . . September 2012

## Safety Symbols

Instruction manual symbol affixed to product. Indicates that the user must refer to the manual for specific WARNING or CAUTION information to avoid personal injury or damage to the product.

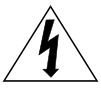
Indicates the field wiring terminal that must be connected to earth ground before operating the equipment — protects against electrical shock in case of fault.




Frame or chassis ground terminal—typically connects to the equipment's metal frame.

Alternating current (AC)

Direct current (DC).



Warning. Risk of electrical shock.

**WARNING** Calls attention to a procedure, practice, or condition that could cause bodily injury or death.

**CAUTION** Calls attention to a procedure, practice, or condition that could possibly cause damage to equipment or permanent loss of data.

## WARNINGS

The following general safety precautions must be observed during all phases of operation, service, and repair of this product. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the product. Agilent Technologies assumes no liability for the customer's failure to comply with these requirements.

**Ground the equipment:** For Safety Class 1 equipment (equipment having a protective earth terminal), an uninterruptible safety earth ground must be provided from the mains power source to the product input wiring terminals or supplied power cable.

**DO NOT operate the product in an explosive atmosphere or in the presence of flammable gases or fumes.**

For continued protection against fire, replace the line fuse(s) only with fuse(s) of the same voltage and current rating and type. DO NOT use repaired fuses or short-circuited fuse holders.

**Keep away from live circuits:** Operating personnel must not remove equipment covers or shields. Procedures involving the removal of covers or shields are for use by service-trained personnel only. Under certain conditions, dangerous voltages may exist even with the equipment switched off. To avoid dangerous electrical shock, DO NOT perform procedures involving cover or shield removal unless you are qualified to do so.

**DO NOT operate damaged equipment:** Whenever it is possible that the safety protection features built into this product have been impaired, either through physical damage, excessive moisture, or any other reason, REMOVE POWER and do not use the product until safe operation can be verified by service-trained personnel. If necessary, return the product to Agilent for service and repair to ensure that safety features are maintained.

**DO NOT service or adjust alone:** Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.

**DO NOT substitute parts or modify equipment:** Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification to the product. Return the product to Agilent for service and repair to ensure that safety features are maintained.

# Declaration of Conformity

Declarations of Conformity for this product and for other Agilent products may be downloaded from the Internet. There are two methods to obtain the Declaration of Conformity:

- Go to http://regulations.corporate.agilent.com/DoC/search.htm . You can then search by product number to find the latest Declaration of Conformity.
- Alternately, you can go to the product web page (www.agilent.com/find/E1468A), click on the Document Library tab then scroll down until you find the Declaration of Conformity link.

## *Notes:*

# Chapter 1
# Getting Started

## Using This Chapter

This chapter gives guidelines to get started using the E1468A and E1469A Relay Matrix Switch modules (Relay Matrix Switches), including:


- Relay Matrix Switches Description . . . . . .11
- Configuring the Relay Matrix Switches . . . . . .15
- Configuring the Terminal Modules . . . . . .20
- Programming the Relay Matrix Switches . . . . . .28


## Relay Matrix Switches Description

The E1468A and E1469A Relay Matrix Switch modules are VXIbus C-Size register-based modules and operate with an E1406 Command Module. Each Relay Matrix Switch consists of a *component module* with 64 two-wire relays and a *terminal module* for connecting user inputs. The component module (E1468-66202) is the same for the E1468A and E1469A. The terminal module for the E1468A (E1468-90011) and the terminal module for the E1469A (E1469-80011) are different for the two Relay Matrix Switch modules.

### E1468A Switch Description

The E1468A Relay Matrix Switch module provides an 8 x 8 two-wire crosspoint matrix. Multiple modules can be wired together creating 8 x 16 (two modules), 16 x 16 (four modules), 8 x 24 (three modules), or larger matrices. Figure 1-1 shows a simplified schematic of the E1468A component module and terminal module.

### E1469A Switch Description

The E1469A Relay Matrix Switch module provides a 4 x 16 two-wire crosspoint matrix. Multiple modules can be wired together creating 4 x 32 (two modules), 8 x 16 (two modules), 4 x 48 (three modules), or larger matrices. Figure 1-2 shows a simplified schematic of the E1469A component module and terminal module.

### E1468A/E1469A Connector Pin-Outs

Each Relay Matrix Switch module consists of a component module and a terminal module. Figure 1-3 illustrates the front panel of an E1468A/E1469A component module and the connector pin-out. The terminal module makes the row and column connection to form the matrix configuration (see Figures 1-1 and 1-2).

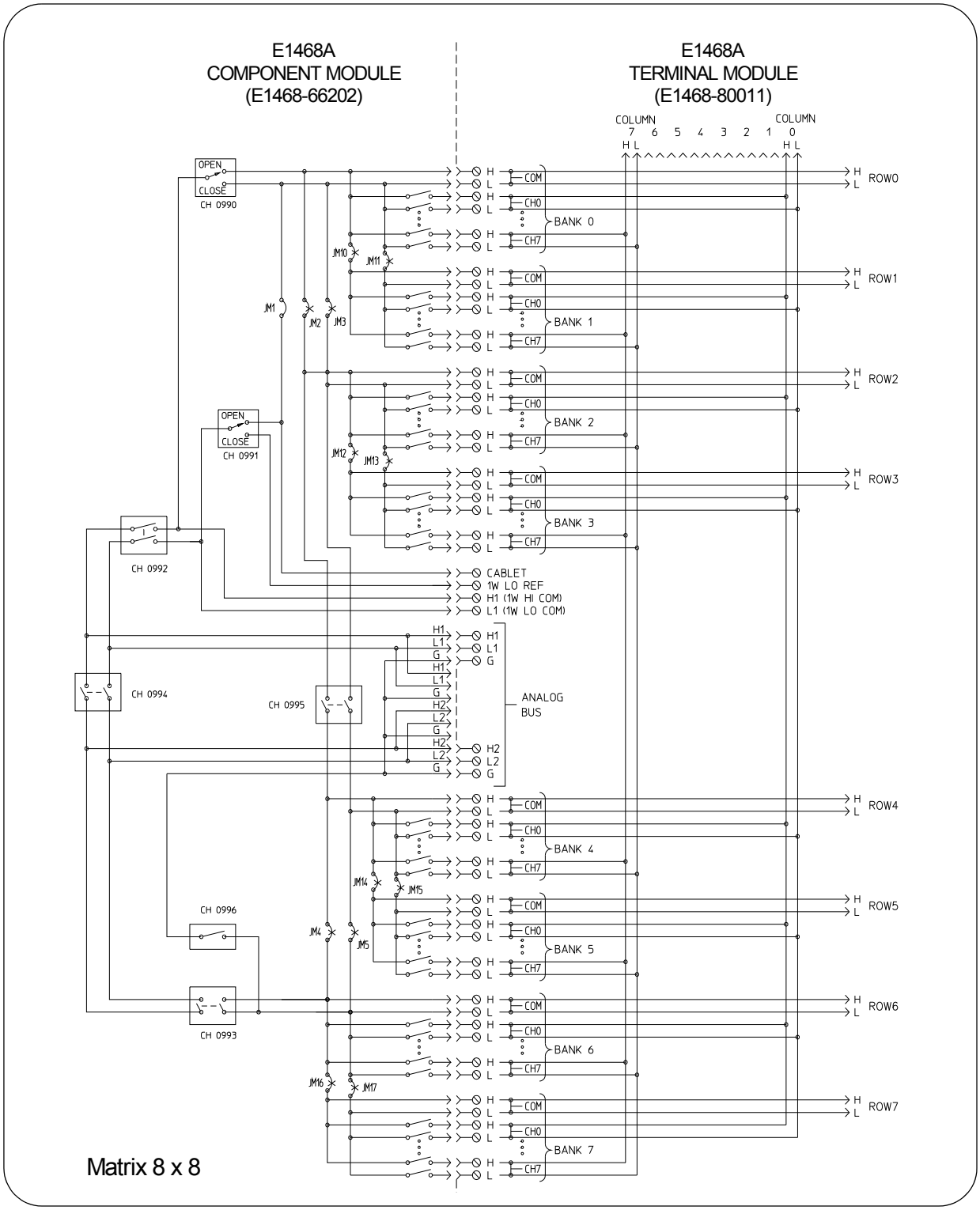



**Figure 1-1. E1468A Simplified Diagram**

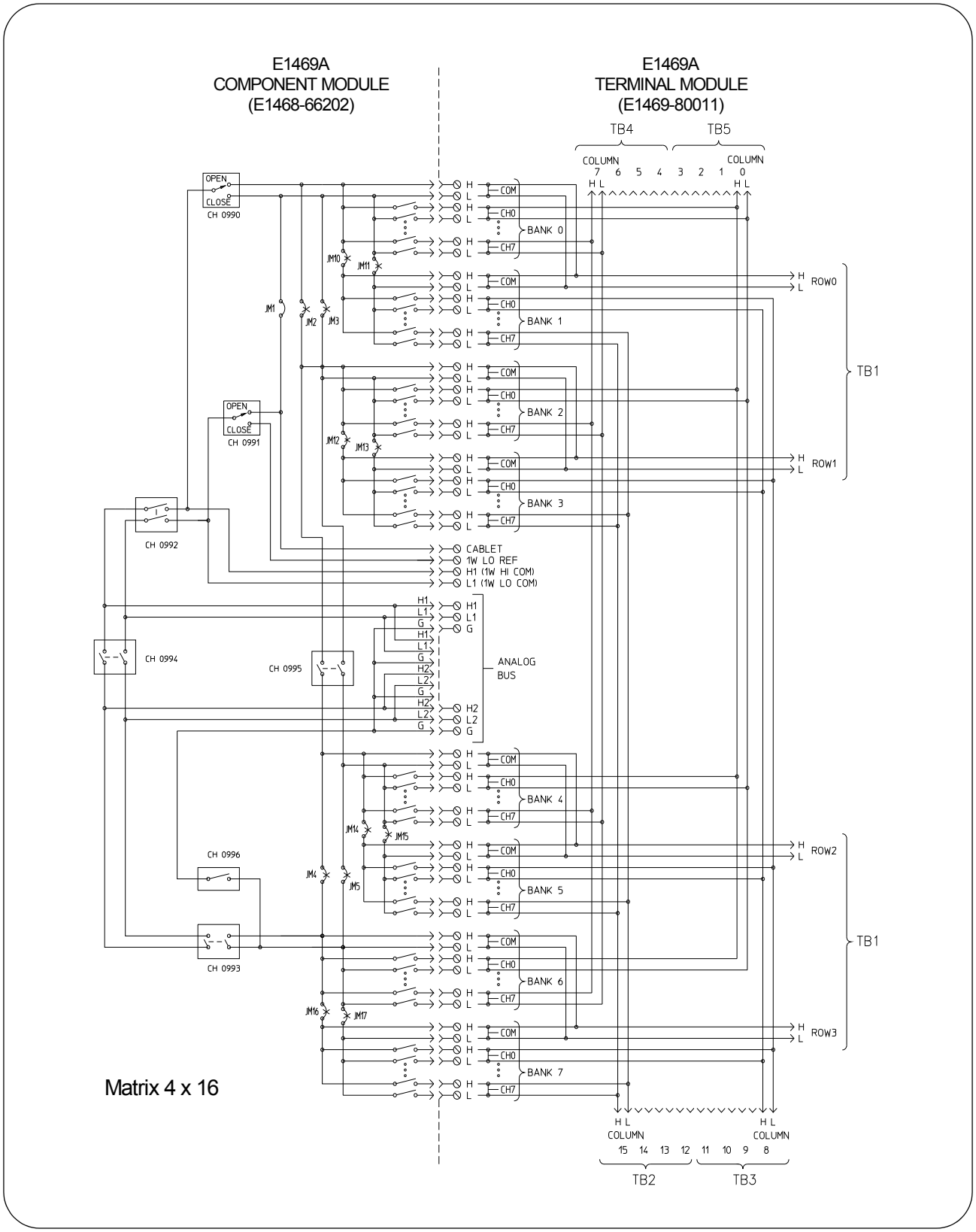



**Figure 1-2. E1469A Simplified Schematic**

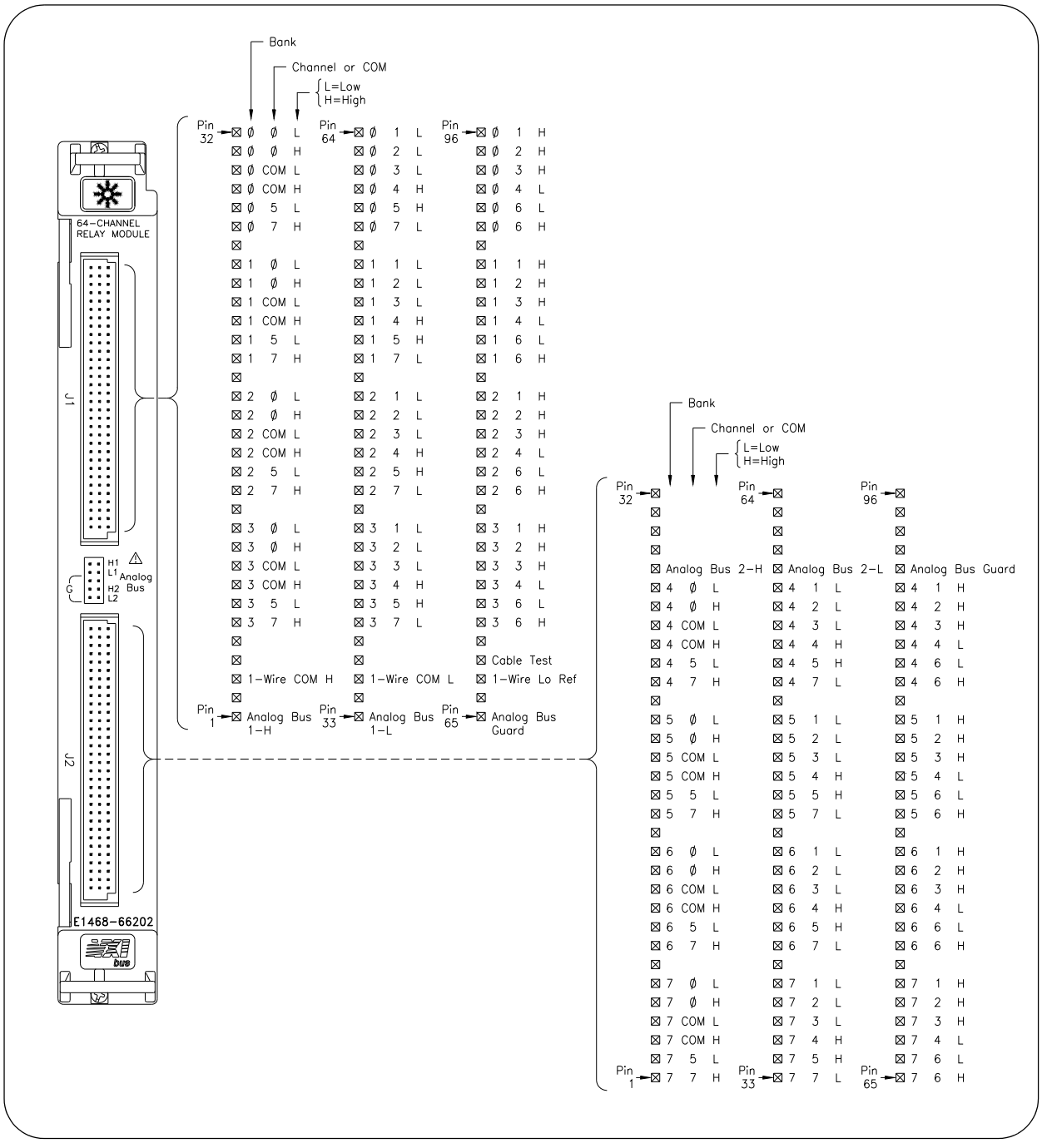



**Figure 1-3. E1468A/E1469A Connector Pin-Out**

# Configuring the Relay Matrix Switches

This section gives guidelines to configure the Relay Matrix Switch modules, including:

- Warnings and Cautions
- Setting the Logical Address Switch
- Setting the Status Register Switch
- Setting the Interrupt Priority
- Installing Relay Matrix Switches in a Mainframe

## Warnings and Cautions

**WARNING** **SHOCK HAZARD. Only service-trained personnel who are aware of the hazards involved should install, remove, or configure the Relay Matrix Switch modules. Before removing any installed module, disconnect AC power from the VXI mainframe and from any devices connected to the Relay Matrix Wwitch modules.**

**WARNING** **CHANNEL WIRING INSULATION. All channels that have a common connection must be insulated so that the user is protected from electrical shock in the event that two or more channels are connected together.**

**CAUTION** **Maximum Inputs.** The maximum voltage that can be applied to any terminal is 220 Vdc/250 Vrms. The maximum current that can be applied to any terminal is 1A at 30 Vdc/Vrms, or 0.3A at 220 Vdc/250 Vrms. The maximum power that can be applied to any terminal is 40 VA.

**CAUTION** **Static Electricity.** Static electricity is a major cause of component failure. To prevent damage to the electrical components in a Relay Matrix Switch module, observe anti-static techniques when removing or installing the module or when working on the module.

## Setting the Logical Address Switch

The logical address switch (LADDR) factory setting is 112. Valid addresses are from 1 to 255. See Figure 1-4 for switch information. The address switch value must be a multiple of 8 if the module is the first module in a "switchbox" used with a VXIbus command module using SCPI commands.

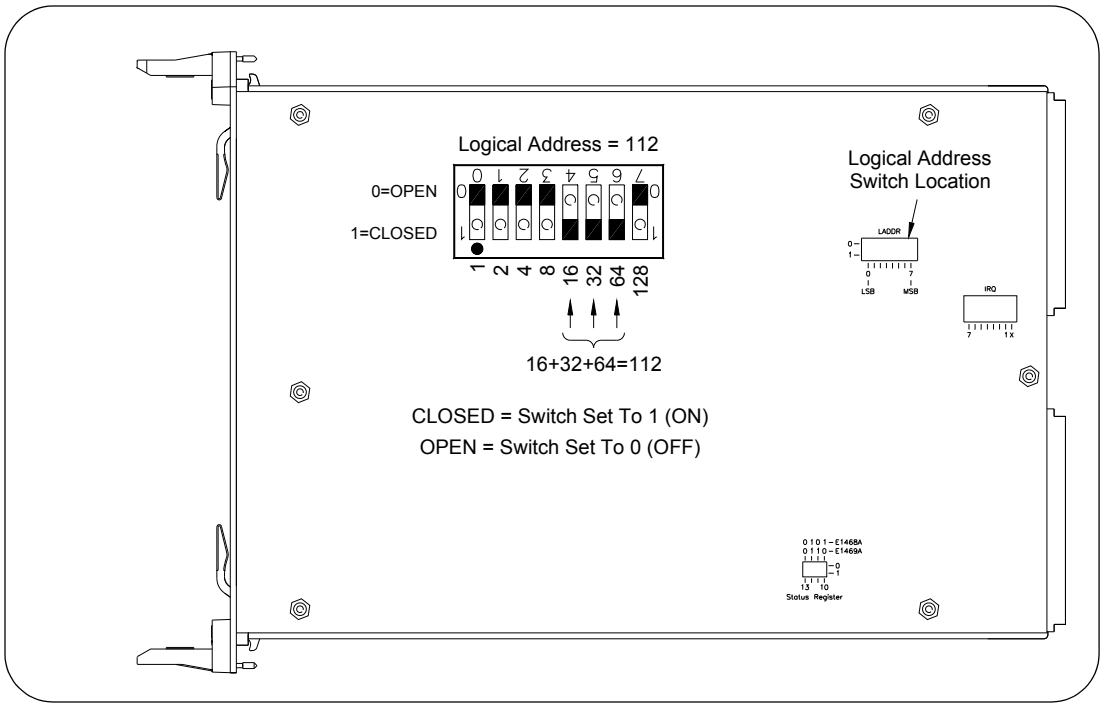



**Figure 1-4. Setting the Logical Address Switch**

## Setting the Status Register Switch

Four bits of the status register switch (bits 10-13) define whether the relay matrix switch module is an E1468A or E1469A. These bits are set automatically when the terminal module is installed.

To ensure proper operation, even without the terminal module, set the status register switch as shown in Figure 1-5. However, if the status register switch is set for the E1468A, but the terminal module is an E1469A (or vice-versa), the interface will not be able to correctly identify and an error will occur.

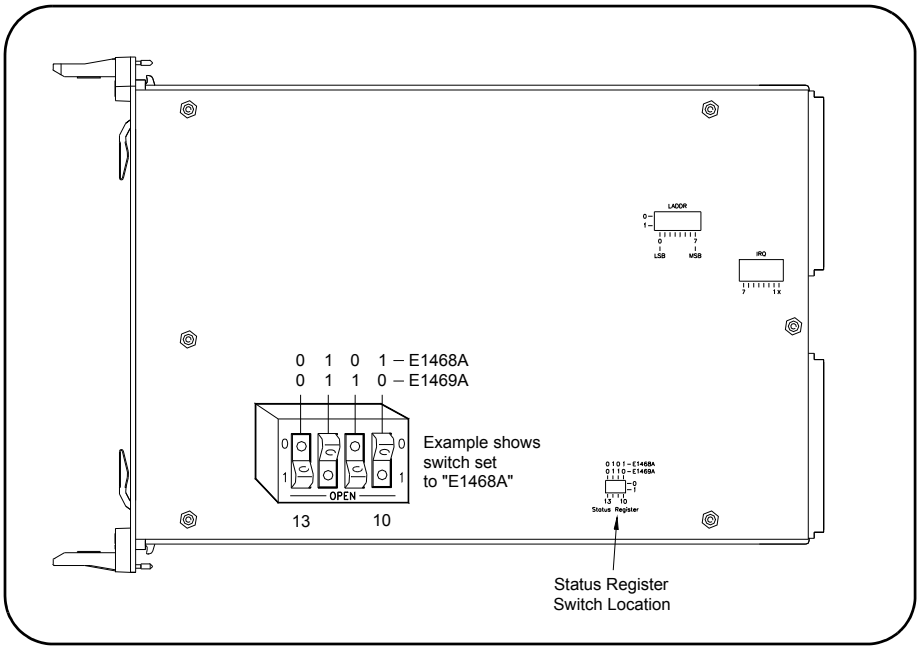



**Figure 1-5. Setting the Status Register Switch**

## Setting the Interrupt Priority

The E1468A/E1469A Relay Matrix Switch modules generate an interrupt after a channel has been closed. These interrupts are sent to, and acknowledgments are received from, the command module (such as an E1406) through the VXIbus backplane interrupt lines.

For most applications where the relay matrix switch module is installed in a C-Size VXI mainframe, the interrupt priority jumper does not have to be moved. This is because the VXIbus interrupt lines have the same priority and interrupt priority is established by installing the modules in slots numerically closest to the E1406 Command Module. Thus, slot 1 has a higher priority than slot 2, slot 2 has a higher priority than slot 3, etc..

See Figure 1-6 to change the interrupt priority. You can select eight different interrupt priority levels. Level 1 is the lowest priority and level 7 is the highest priority. Level X disables the interrupt.

The module's factory setting is level 1. To change the priority level, remove the four-pin jumper from the old priority location and reinstall the jumper in the new priority location. If the four-pin jumper is not used, the two jumper locations must have the same interrupt priority level selected.

**NOTE** *The interrupt priority jumper must be installed in position 1 when using the E1406 Command Module. Level X interrupt priority should not be used under normal operating conditions. Changing the interrupt priority level jumper is not recommended.*

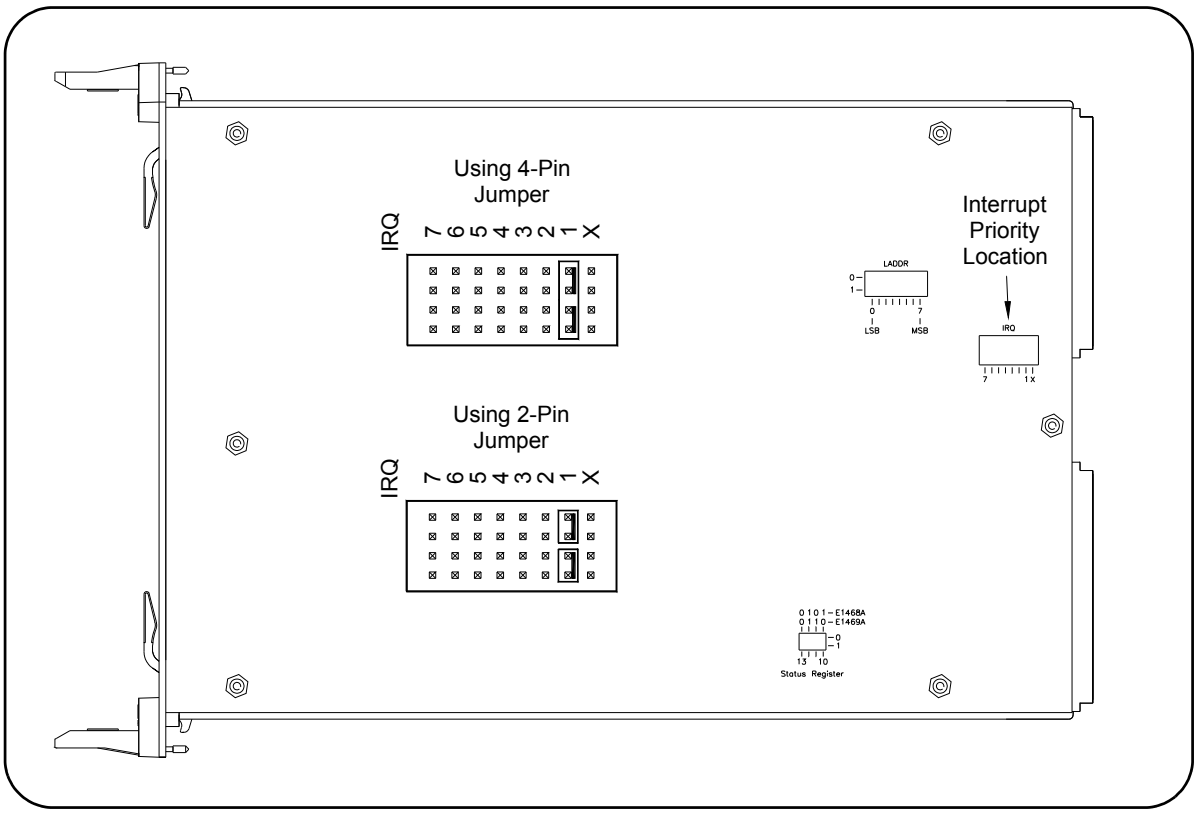



**Figure 1-6. Interrupt Priority Selection**

## Installing Relay Matrix Switches in a Mainframe

The E1468A/E1469A modules may be installed in any slot (except slot 0) in a C-Size VXI mainframe. See Figure 1-7 to install a module in a mainframe.

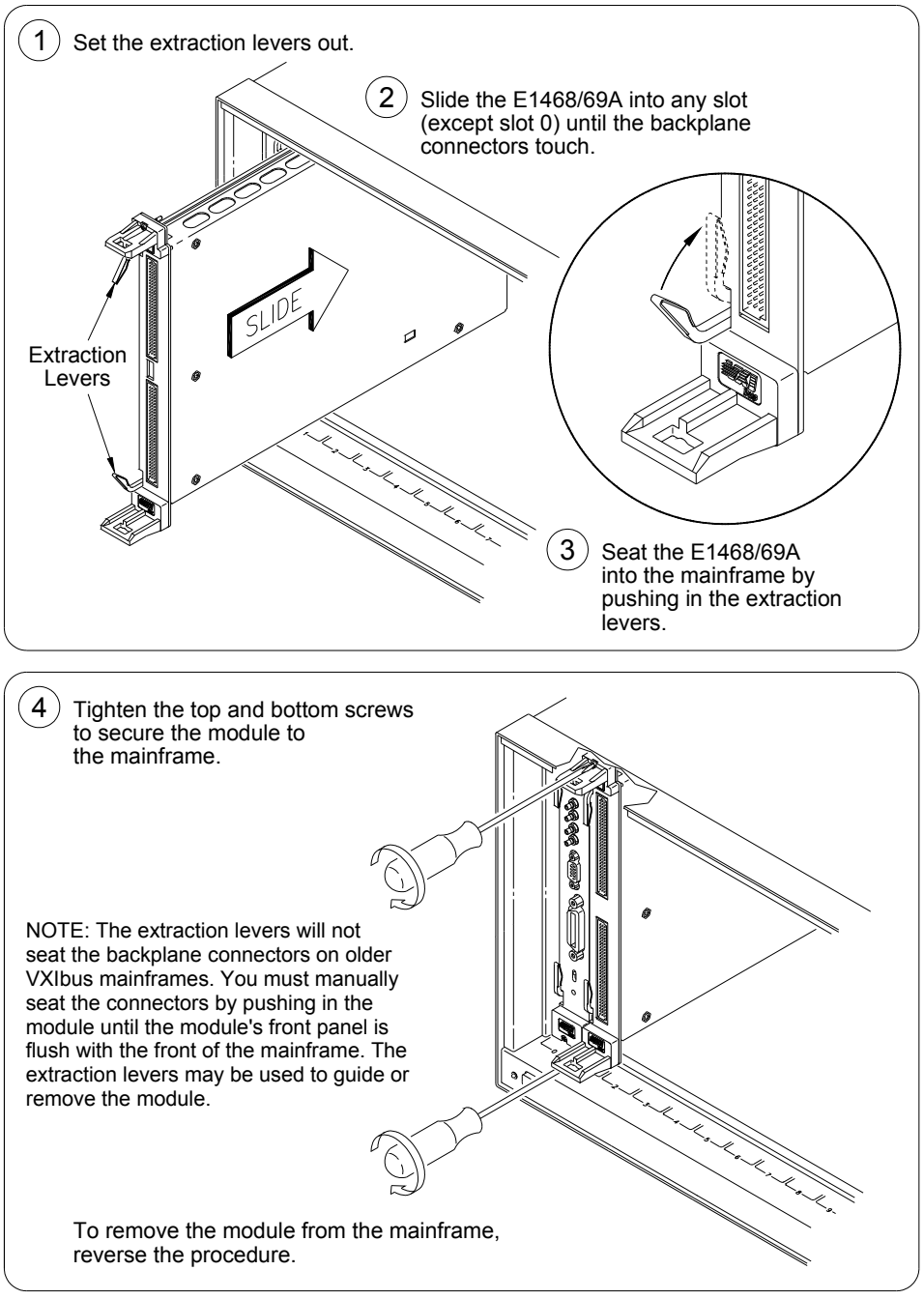



**Figure 1-7. Installing Relay Matrix Switches in a Mainframe**

# Configuring the Terminal Modules

This section gives guidelines to configure the E1468A and E1469A terminal modules, including:

- Wiring the Terminal Module
- Creating Larger Matrixes
- Attaching a Terminal Module to the Relay Switch Module

## Wiring the Terminal Modules

Guidelines to wire the E1468A and E1469A terminal modules follow.

### E1468A Terminal Module Connectors

Figure 1-8 shows the E1468A terminal module connectors and associated row/column designators. Shielding jumpers JM1 - JM10 are shown. See "Creating Larger Matrices" for information on using the expansion connectors J1 - J4 and for shield wiring details.

**NOTE** *Jumpers JM1 - JM10 on the E1468A terminal module connect row/column shields to earth ground through the VXIbus backplane. You may want to remove one or more of these jumpers to reduce common mode noise.*

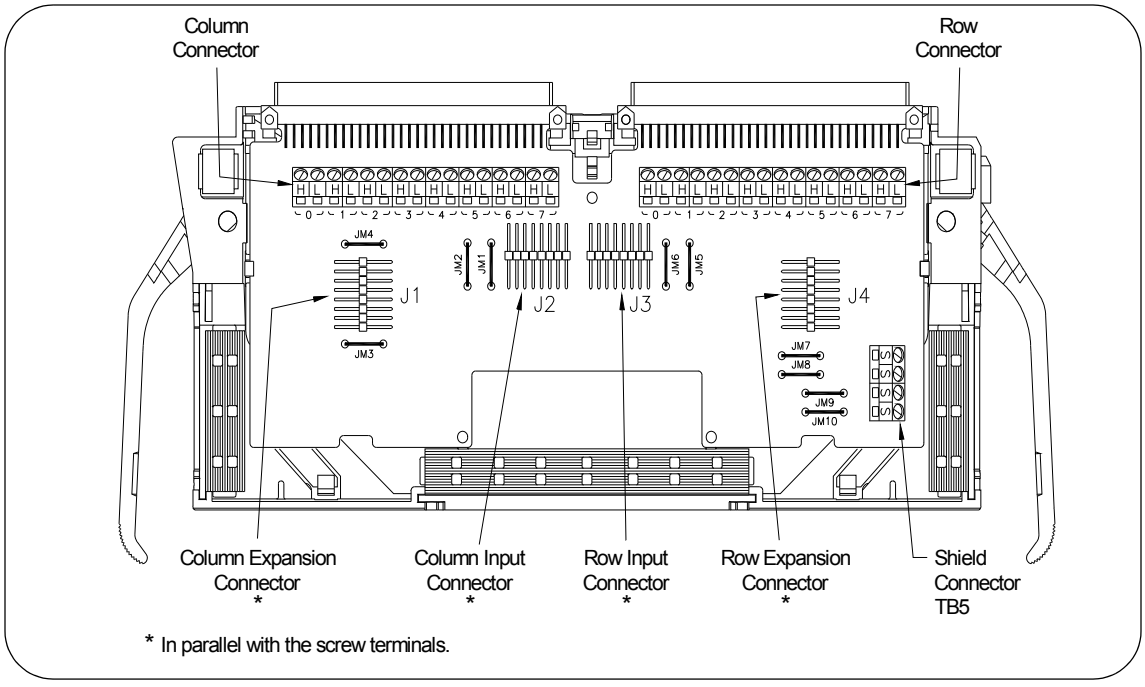



**Figure 1-8. E1468A 8 x 8 Matrix Switch Terminal Module**

## E1469A Terminal Module Connectors

Figure 1-9 shows the E1469A terminal module connectors and associated row/column designators. Shielding jumpers JM1 - JM12 are shown. See "Creating Larger Matrices" for information on using the expansion connectors J1 - J5 and for shield wiring details.

**NOTE** *Jumpers JM1 - JM12 on the E1469A terminal module connect row/column shields to earth ground through the VXIbus backplane. You may want to remove one or more of these jumpers to reduce common mode noise.*

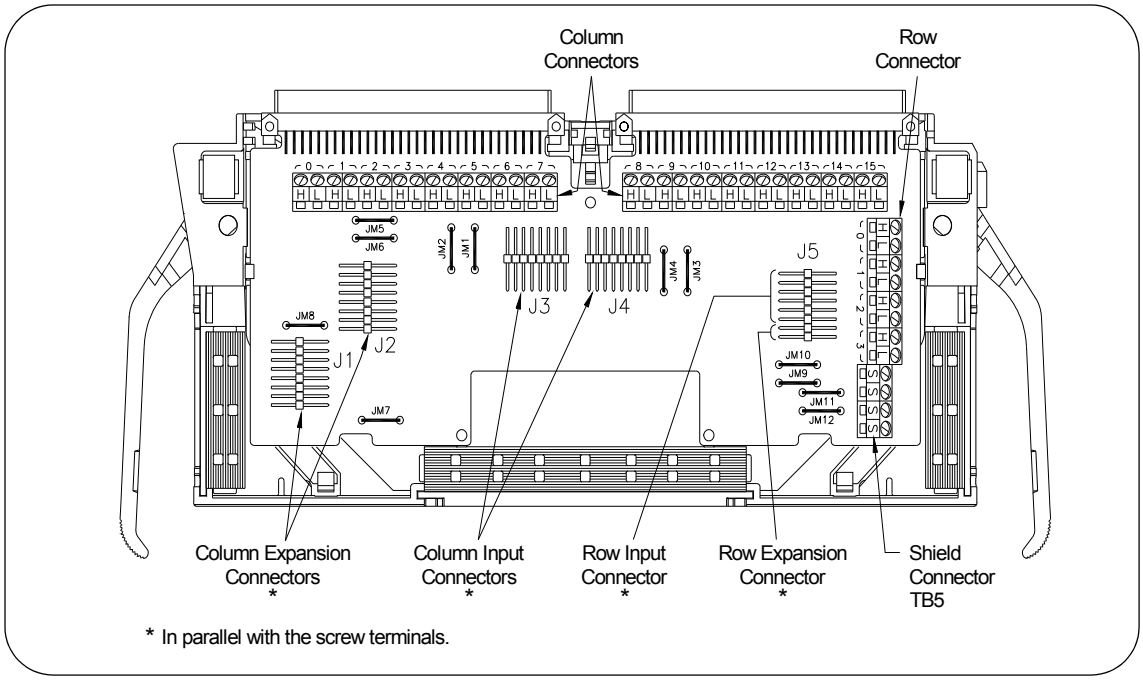



**Figure 1-9. E1469A 4 x 16 Matrix Switch Terminal Block**

## Available Cables

To assist you in wiring Relay Matrix Switch terminal modules into your test system, this table shows a list of cables that are available from Agilent.

| Description | Finished Length | End "A" | End "B" | Part Number |
|---|---|---|---|---|
| Module expansion connector with quick disconnect (twisted pair) | ~30 cm | 4 x 2 connector for expansion connectors on terminal modules | 4 x 2 connector for expansion connectors on terminal modules | E1468-80002 |
| 50 Ω Coax | 2.0 m | 2-pin TLA* | BNC (molded over) | E1065-61620 |
| Dual banana instrument | 2.0 m | 3-pin TLA* | Dual banana | E1066-61620 |
| SMB instrument | 2.0 m | 2-pin TLA* | SMB (molded over) | E1068-61620 |

*TLA is a family of connector/cable assemblies with good transmission line design that are made by an Agilent supplier. The 2-pin and 3-pin TLA connectors are designed to fit on one channel of the terminal module expansion connectors.

## Terminal Module Wiring Guidelines

User wiring to the Relay Matrix Switch modules is to the High (H) and Low (L) connections on terminal module. Figure 1-10 gives guidelines to wire the terminal modules. Maximum terminal wire size is No. 16 AWG. Wire ends should be stripped 6mm (0.25 in.) and tinned. When wiring all channels, use a smaller gauge wire (No. 20 - 22 AWG). The expansion connectors allow you to create larger matrices. See "Creating Larger Matrices".

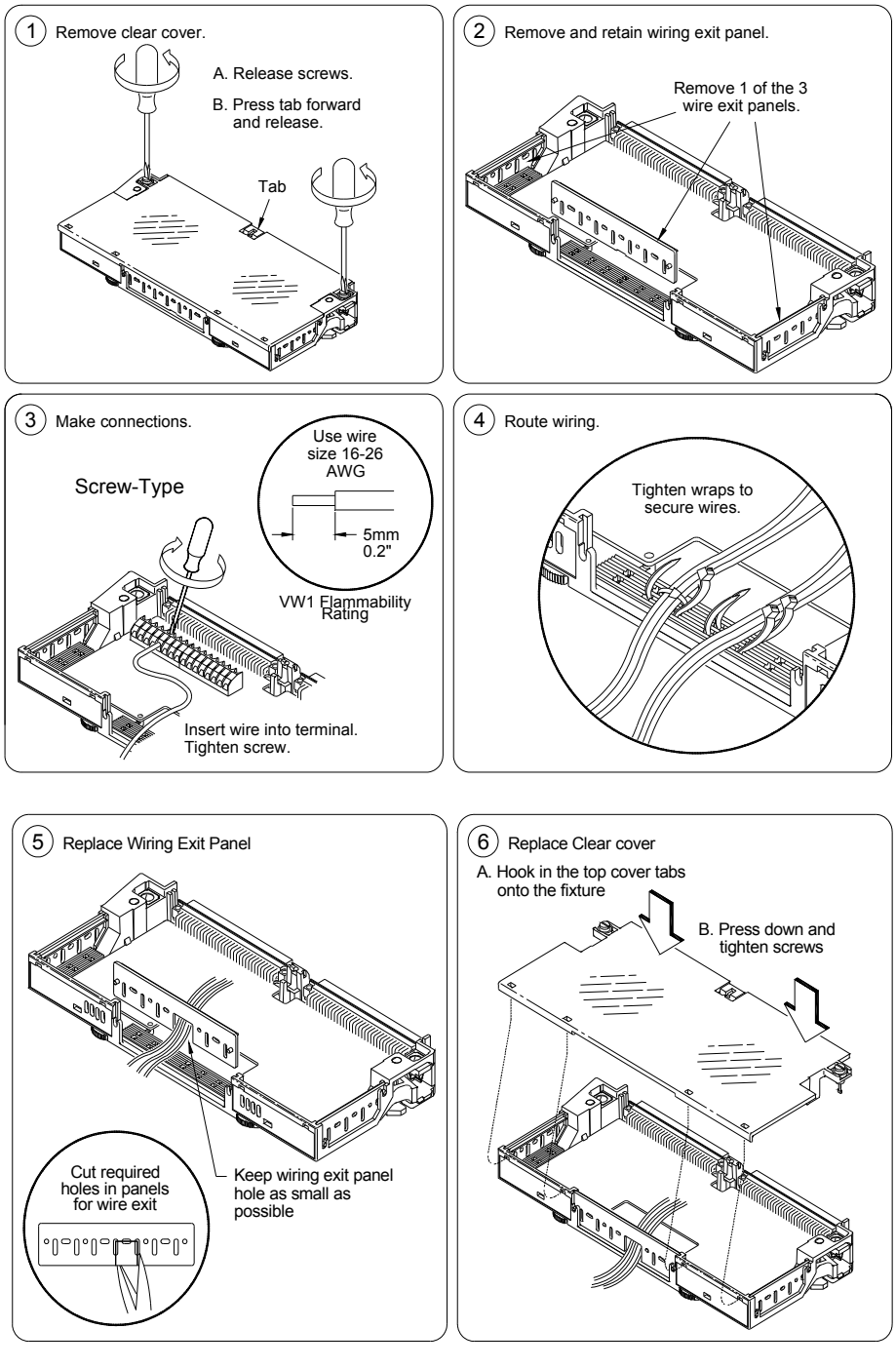



**Figure 1-10. Wiring the Terminal Module**

## Creating Larger Matrixes

You can use the expansion connectors on the terminal module to interconnect modules to create larger matrixes. Use part number E1468-80002 Daisy-Chain Cable (a 4-pair High and Low cable assembly) for expansion between modules. This cable provides a quick-disconnect allowing easy removal of modules.

## Shield Wiring Details

Figure 1-11 shows shield wiring details for the E1468A and E1469A terminal modules.

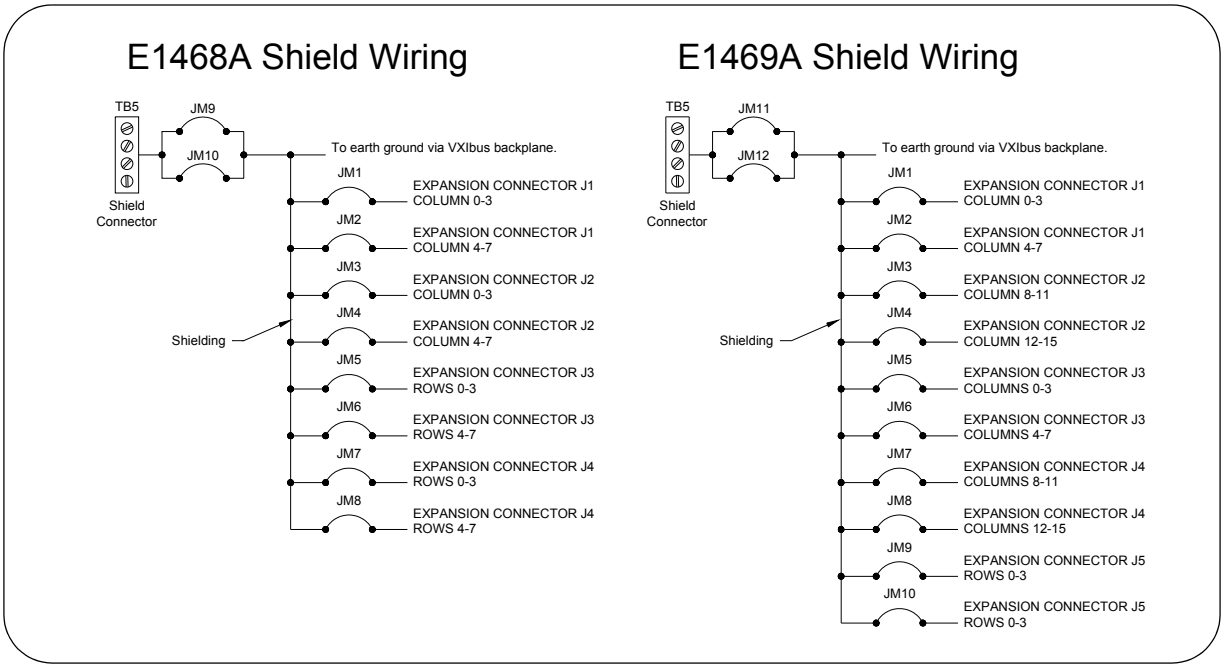



**Figure 1-11. E1468A and E1469A Terminal Module Shield Wiring**

## 8 x 24 Matrix

Figure 1-12 shows how to connect three E1468A Relay Matrix Switch Modules to create an 8-row by 24-column matrix. This configuration requires four E1468-80002 Daisy-Chain Cables.

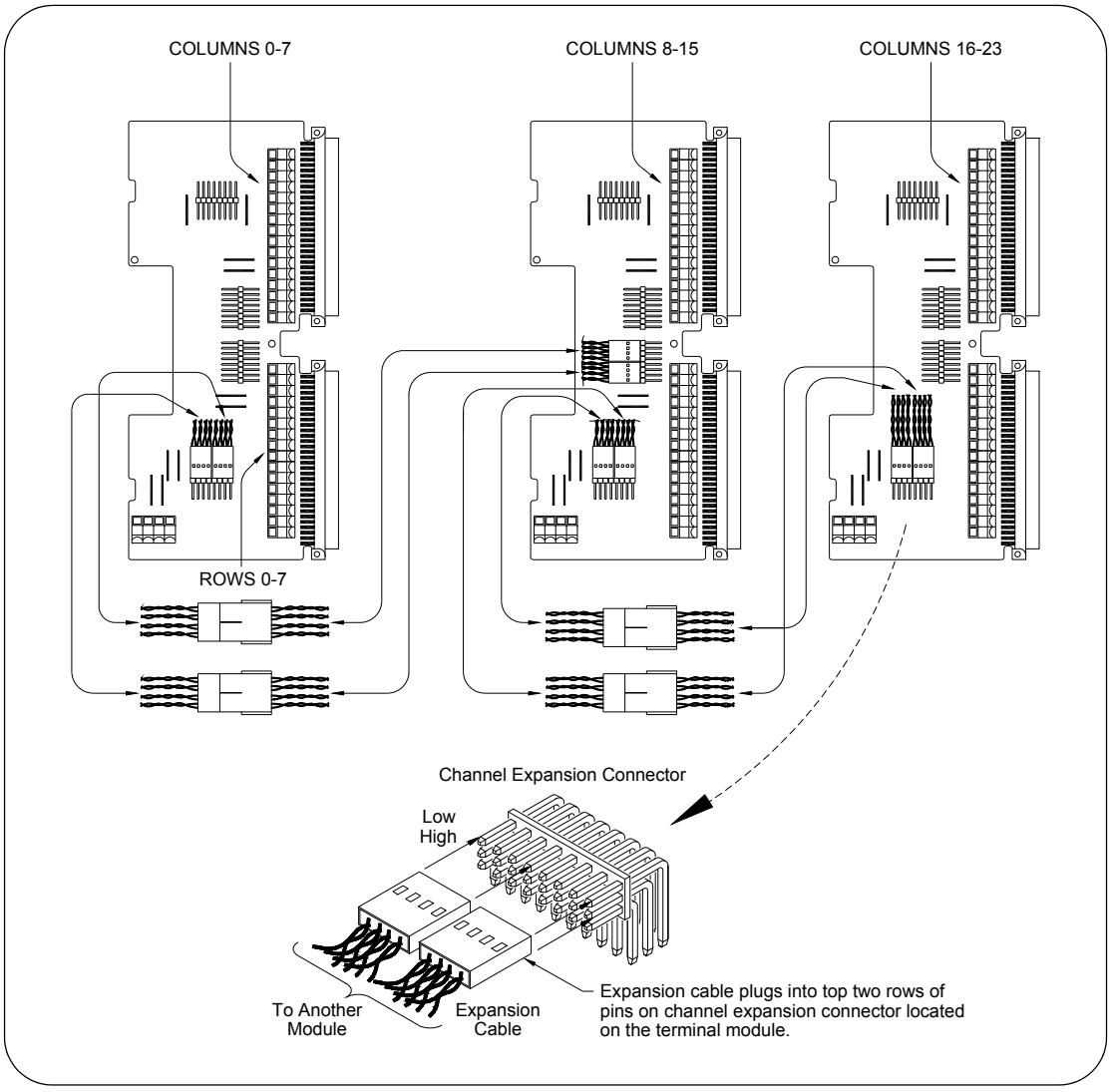



**Figure 1-12. 8-Row x 24-Column Matrix Using E1468A Terminal Module**

## 16 x 16 Matrix

Figure 1-13 shows how to connect four E1468A Relay Matrix Switch Modules to create a 16-row by 16-column matrix. This configuration requires eight E1468-80002 Daisy-Chain Cables.

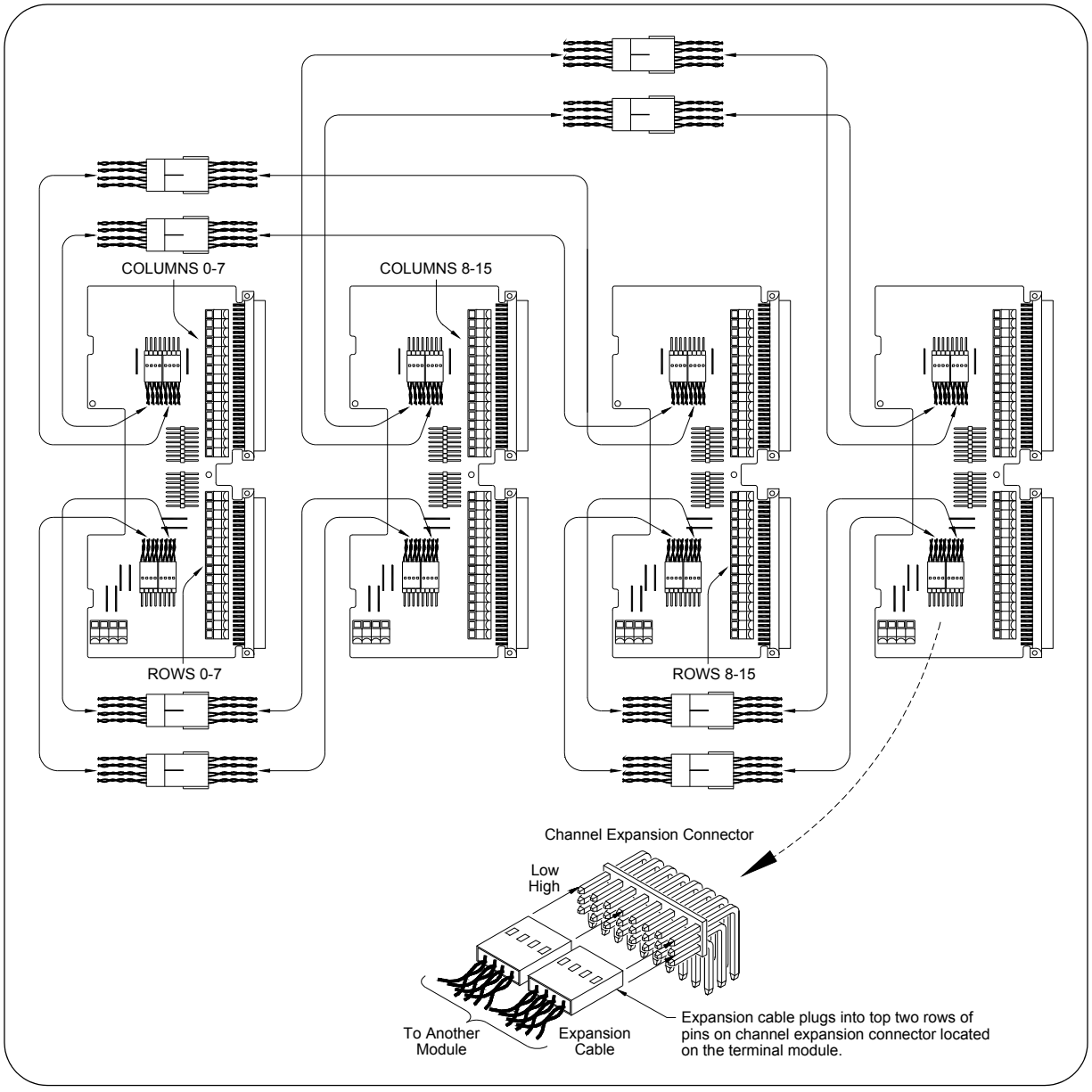



**Figure 1-13. 16-Row x 16-Column Matrix Using E1468A Terminal Module**

## 4 x 48 Matrix

Figure 1-14 shows how to connect three E1469A Relay Matrix Switch Modules to create a 4-row by 48-column matrix. This configuration requires two E1468-80002 Daisy-Chain Cables.

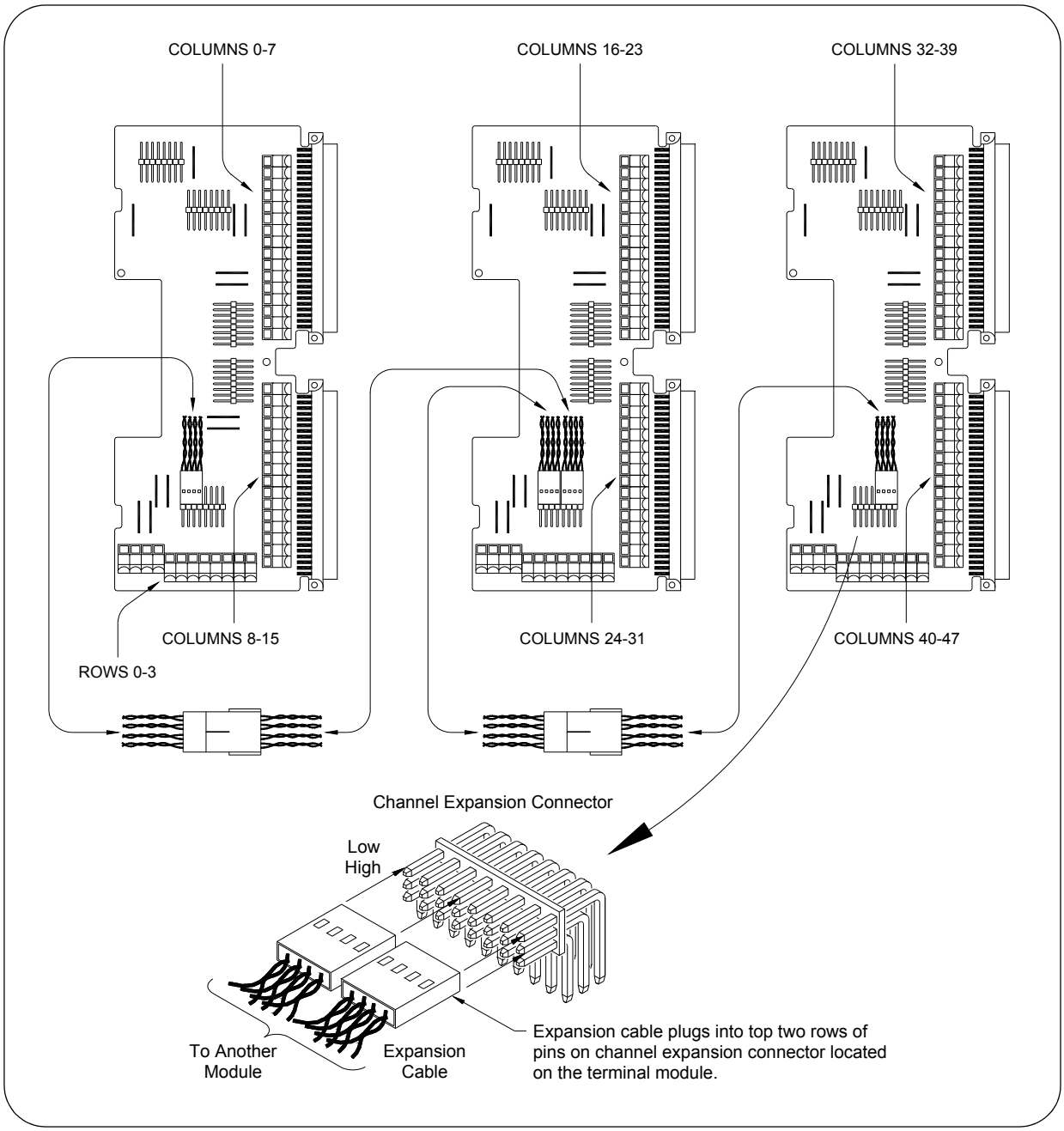



**Figure 1-14. 4-Row x 48-Column Matrix Using E1469A Terminal Block**

## Attaching a Terminal Module to the Relay Switch Module

Figure 1-15 gives guidelines to attach a terminal module to a component module.

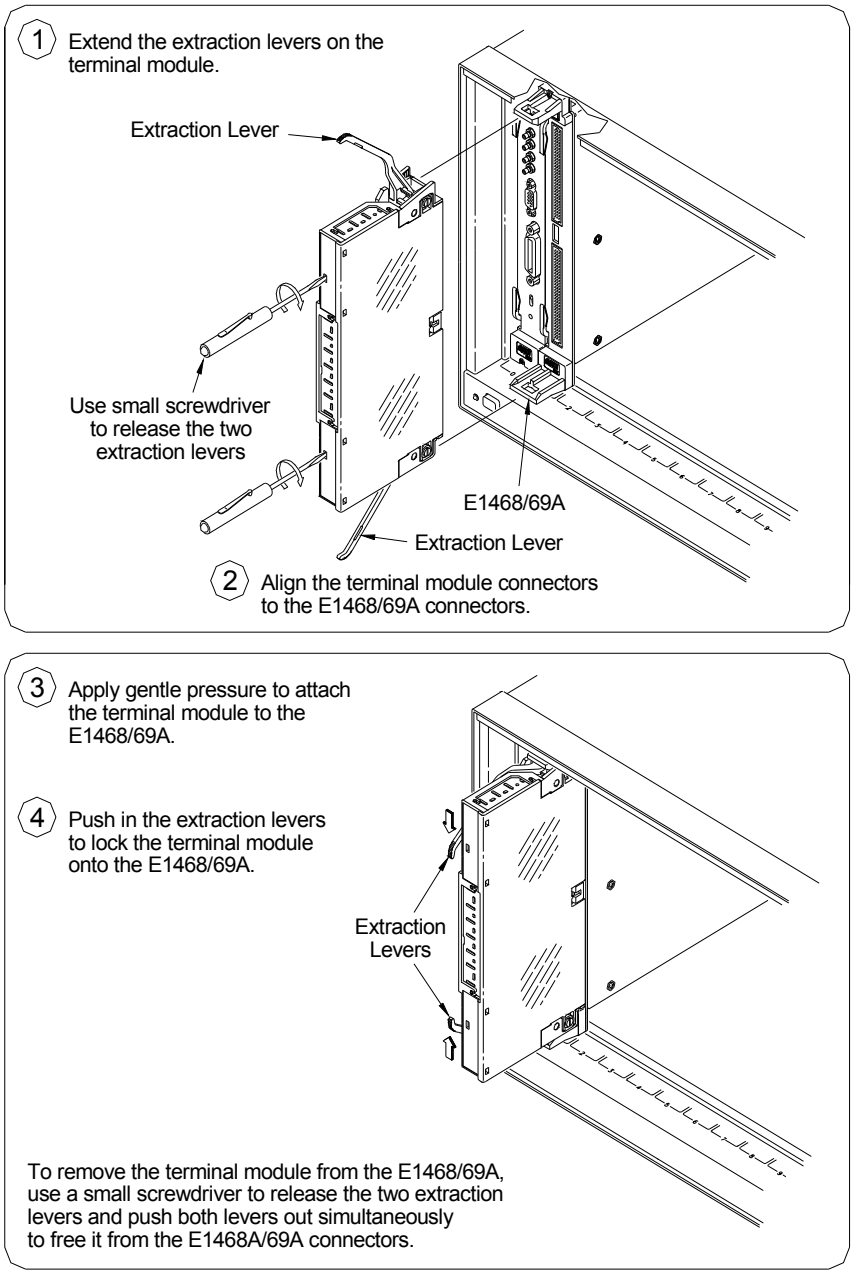



**Figure 1-15. Attaching a Terminal Module to the Relay Matrix Switch Module**

# Programming the Relay Matrix Switches

This section gives guidelines to program the Relay Matrix Switches, including:

- Using SCPI Commands
- Addressing the Modules
- Initial Operation

## Using SCPI Commands

VXIbus plug-in modules installed in a C-Size VXI mainframe are treated as independent *instruments* having a unique secondary GPIB address. Each instrument is also assigned a dedicated error queue, input and output buffers, status registers, and, if applicable, dedicated mainframe memory space for readings or data. An instrument may be composed of a single plug-in module (such as a counter) or multiple plug-in modules (for a switchbox or scanning voltmeter instrument).

To program the Relay Matrix Switch module using Standard Commands for Programmable Instruments (SCPI), you must select the computer language, interface address, and SCPI commands to be used. Guidelines to select SCPI commands for the relay matrix switch module follow.

**NOTE** *This discussion applies only to SCPI programming. See Appendix B for information on Relay Matrix Switch registers.*

## Addressing the Modules

To address specific channels (relays) within a relay matrix, you must specify the SCPI command and the Relay Matrix Switch channel address. Use CLOSe <*channel_list*> to close specified relay(s), OPEN <*channel_list*> to open specified relay(s), and SCAN <*channel_list*> to close the set of relays specified.

### Module Card Numbers

The matrix card (module) number depends on the switchbox configuration (single-module or multiple-module) set for the matrices. (Leading zeroes can be ignored for the card number.) For a single-module switchbox, the card number is always 01.

For a multiple-module switchbox, the card numbers are 01, 02,...nn. The module with the lowest logical address is card number 01, the module with the next-lowest logical address is card number 02, etc..

### E1468A Relay Matrix Switch Channel Addresses

For the E1468A Relay Matrix Switch module, the channel address (*channel_list*) has the form (@*ssrc*) where *ss* = card number (01-99), *r* = row number, and *c* = column number. E1468A Relay Matrix Switch module channel numbers are *r* = 0 to 7 (one digit) and *c* = 0 to 7 (one digit).

You can address single channels (@*ssrc*); multiple channels (@*ssrc,ssrc,...*); sequential channels (@*ssrc:ssrc*); groups of sequential channels; @*ssrc:ssrc,ssrc:ssrc*); or any combination. For example, CLOS (@124) closes row 2, column 4 of card 01 of an E1468A Relay Matrix Switch module.

Only valid channels can be accessed in a channel list or channel range. Also, the channel list or channel range must be from a lower channel number to a higher channel number. For example, CLOS (@100:233) is acceptable, but CLOS (@233:100) generates an error.

### E1469A Relay Matrix Switch Channel Addresses

For the E1469A Relay Matrix Switch module, the channel address (*channel_list*) has the form (@*ssrrcc*) where *ss* = card number (01-99), *rr* = row number, and *cc* = column number. E1469A 4 x 16 Relay Matrix Switch module channel numbers are *rr* = 00 to 03 (two digits) and *cc* = 00 to 15 (two digits).

You can address single channels (@*ssrrcc*); multiple channels @*ssrrcc,ssrrcc,...*); sequential channels (@*ssrrcc:ssrrcc*); groups of sequential channels (@*ssrrcc:ssrrcc,ssrrcc:ssrrcc*); or any combination. For example, CLOS (@10214) closes row 02, column 14 of card 01 of an E1469A Relay Matrix Switch module.

Only valid channels can be accessed in a channel list or channel range. Also, the channel list or channel range must be from a lower channel number to a higher channel number. For example, CLOS (@10000:20303) is acceptable, but CLOS (@20303:10000) generates an error.

## Initial Operation

An example program follows that uses BASIC and SCPI language to help get you started using the Relay Matrix Switch modules. The example assumes a GPIB interface. The program closes row 03, column 12 of an E1469A 4 x 16 Relay Matrix Switch module at logical address 112 (secondary address = 112/8 = 14) and queries the result. The result is returned to the controller and displayed (1 = relay closed, 0 = relay open).

```
10 OUTPUT 70914; "*RST"                    !Reset the module. Set
                                           !all relays to open.
20 OUTPUT 70914; "CLOS (@10312)"           !Close channel row 03,
                                           !column 12 on the first
                                           module in the switchbox
30 OUTPUT 70914; "CLOS? (@10312)"          !Query channel
40 ENTER 70914; Value                      !Enter result
50 PRINT Value                             !Print results
60 END
```

## *Notes:*

# Chapter 2
# Using the Relay Matrix Switches

## Using This Chapter

This chapter uses typical examples to show how to use the Relay Matrix Switch modules. It contains the following sections:


- Relay Matrix Switch Commands/States . . . . . .31
- Relay Matrix Switch Functions . . . . . .33


**NOTE** *All examples in this chapter use GPIB select code 7, primary address 09, and secondary address 14 (LADDR = 112) for the modules.*

## Relay Matrix Switch Commands/States

This section shows the relay matrix commands used in this chapter, the query commands, and the power-on/reset states.

### Relay Matrix Switch Commands

This table shows some of the commands used in this chapter. Commands in square brackets ([ ]) are implied and are not sent with the command. See Chapter 3 for additional information.

| Command | Description |
|---|---|
| INITiate[:IMMediate] | Starts the scan sequence and closes the first channel in the *channel_list*. |
| [ROUTe:]CLOSe <*channel_list*> | Closes the channels in the *channel_list*. |
| [ROUTe:]CLOSe? <*channel_list*> | Queries the state of the channels in the *channel_ list*. |
| [ROUTe:]OPEN <*channel_list*> | Opens the channels in the *channel_list*. |
| [ROUTe:]OPEN? <*channel_list*> | Queries the state of channels in the *channel_ list*. |
| [ROUTe:]SCAN <*channel_list*> | Defines the *channel_list* to be scanned. Channels specified are closed one at a time. |
| TRIGger:SOURce <*source*><br>*source* = BUS \| EXT \| HOLD \| IMM \| TTLT \| ECLT | Selects the trigger *source* to advance the scan. |
| *CLS | Clears switchbox status registers and error queue. |
| *RST | Resets the hardware to a known state. |

## Relay Matrix Switch Query Commands

All query commands end with a "?". All data is sent to the output buffer where you can retrieve it into your computer. The following are valid query commands:

| Query | Description |
|---|---|
| ARM:COUN? | Number of Scanning Cycles |
| CLOS? | Channel Closed |
| INIT:CONT? | Scanning State |
| OPEN? | Channel Open |
| OUTP:ECLT*n*? | ECL Trigger Output State |
| OUTP:EXT? | External Trigger Output State |
| OUTP:TTLT*n*? | TTL Trigger Output State |
| STAT:OPER:ENAB? | Status Operation Enable |
| STAT:OPER[:EVEN]? | Status Operation Event |
| SYST:CDES? <*number*> | Module Description |
| SYST:CTYP? <*number*> | Module Type |
| SYST:ERR? | System Error |
| TRIG:SOUR? | Trigger Source |

## Power-on and Reset Conditions

When power is first applied to the Relay Matrix Switch modules or *RST (reset) is executed, all relays are open. This table lists the parameters and default values for the switchbox functions described in this chapter. Commands in brackets ( [ ] ) are implied and are not sent with the command.

| Parameter | Default | Description |
|---|---|---|
| ARM:COUNt | 1 | Number of scanning cycles is 1 |
| TRIGger:SOURce | IMM | Will advance scanning cycles automatically |
| INITiate:CONTinuous | OFF | Number of scanning cycles set by ARM:COUNt |
| OUTPut[:EXTernal][:STATe] | OFF | Trigger output from EXTernal, TTLTrg, or ECLTrg sources is disabled |

# Relay Matrix Switch Functions

This section provides some examples for Relay Matrix Switch module functions, including:

- Checking Module Identification
- Switching Channels
- Recalling and Saving States
- Detecting Error Conditions
- Synchronizing Relay Switch Modules

## Checking Module Identification

You can use the *RST, *CLS, *IDN?, CTYP?, and CDES? commands to reset and identify the Relay Matrix Switch modules.

### Example: Identifying Relay Matrix Switch Modules

This program uses the *RST, *CLS, *IDN?, CTYP?, and CDES? commands to reset and identify the Relay Matrix Switch modules.

```
10  DIM A$[50]; B$[50], C$[50]
20  OUTPUT 70914; "*RST; *CLS; *IDN?"
30  ENTER 70914; A$
40  OUTPUT 70914; "SYST:CDES? 1"
50  ENTER 70914; B$
60  OUTPUT 70914; "SYST:CTYP? 1"
70  ENTER 70914; C$
80  PRINT A$
90  PRINT B$
100 PRINT C$
110 END
```

A typical return is:

```
HEWLETT-PACKARD,SWITCHBOX,0,A.04.00
4x16 2-WIRE MATRIX
HEWLETT-PACKARD,E1469A,0,A.04.00
```

## Switching Channels

Use CLOSe <*channel_list*> to close one or more Relay Matrix Switch channels, and OPEN <*channel_list*> to open the channel(s). *channel_list* has the following forms.

For the E1468A only, the form is @*ssrc* where *ss* = card number (01-99) *r* = row number (0 to 7 [one digit]) and *c* = column number (0 to 7 [one digit]). For the E1469A only, the form is @*ssrrcc* where *ss* = card number (01-99) *rr* = row number (00 to 03 [two digits]) and *cc* = column number (00 to 15 [two digits]).

To OPEN or CLOSe multiple channels, place a comma (,) between the channel numbers. For example, to close channels 10103 and 10201, execute CLOS 10103,10201. To OPEN or CLOSe a contiguous range of channels, place a colon (:) between the first and last channel numbers.

### Example: Opening/Closing Rows/Columns

This program shows how to close and open row 2 (02), column 14 on an E1469A Relay Matrix Switch module (card #1):

```
10  DISP "TEST E1469A MATRIX"
20  OUTPUT 70914; "ROUT:CLOS (@10214)"
30  OUTPUT 70914; "ROUT:OPEN (@10214)"
40  END
```

### Example: Sequencing Channels (E1468A)

This program sequences through each channel on an E1468A 8x8 Relay Matrix Switch Module.

```
10 DIM E$[128]
20 FOR I = 0 TO 7
30  FOR J = 0 TO 7
40       A = 100 + 10 * I + J
50            OUTPUT 70914; "ROUT:CLOS (@ ";A;")"
60            OUTPUT 70914; "ROUT:CLOS? (@100:177)"
70       ENTER 70914; E$
80  PRINT "CHANNEL CLOSED NOW"; E$
90       OUTPUT 70914; "ROUT:OPEN (@ ";A;")"
100      NEXT J
110 NEXT I
120 END
```

### Example: Sequencing Channels (E1469A)

To use this program with the E1469A 4x16 Relay Matrix Switch Module, replace lines 20, 30, 40, and 60 with:

```
20 FOR I = 0 TO 3
30  FOR J = 0 TO 15
40       A = 10000 + 100 * I + J
60            OUTPUT 70914; "ROUT:CLOS? (@10000:10315)"
```

## Recalling and Saving States

The *SAV <*numeric_state*> stores the current state of the switchbox channels. Up to 10 states may be stored by specifying the <*numeric_state*> as an integer 0 through 9. The following states are stored:

- Channel relay states (open or closed)
- ARM:COUNt
- TRIGger:SOURce <*source*>
- OUTPut[:EXTernal][:STATe]
- INITiate:CONTinuous

The *RCL <*numeric_state*> command recalls the specified previously stored state. If the specified <*numeric_state*> does not exist, the Relay Matrix Switch module configures to its power-on/reset states.

### Example: Saving and Recalling States

This examples closes channels on the module and saves the state as number 5. When the saved state is recalled, only the channels that were closed in the stored state are closed. All other channels in the switchbox are opened.

```
10  OUTPUT 70914;"CLOS (@10000:10015)"          !Close ch 00 through 15
20  OUTPUT 70914; "*SAV 5"                      !Save as state 5
30  OUTPUT 70914; "*RST; *CLS"                  !Reset and clear status reg
40  OUTPUT 70914; "CLOS (@10113,10112,10200)"   !Close ch 13, 12, 00
50  OUTPUT 70914; "*RCL 5"                      !Recall the stored state.
60  END
```

## Detecting Error Conditions

You can use the SYST:ERR? command to poll the switchbox for errors. You can also use interrupts to signal the controller when an error occurs.

### Example: Illegal Channel Closure Error

This program attempts an illegal channel closure and polls for the error message:

```
10  DIM Err_num$[256]
20  OUTPUT 70914; "CLOS (@10500)"
30  OUTPUT 70914; "SYST:ERR?"
40  ENTER 70914; Err_num$
50  PRINT Err_num$
```

### Example: Using Interrupts to Signal Errors

This program uses an interrupt to signal the controller when an error occurs. The SYST:ERR? command returns the error message.

```
10  ON INTR 7 CALL Errmsg                       !Call subprogram Errmsg if a module
                                                !programming error occurs
20  ENABLE INTR 7:2                             !Enable the computer to respond to the
                                                interrupt from the module
30  OUTPUT 70914; "*SRE 32; *ESE 64"            !Unmask the Event Status bit in the module's
                                                !Status Register (*SRE 32). Unmask the
                                                !module error conditions in its Standard Event
                                                Status Register (*ESE 64).
40  OUTPUT 70914 ".  .  .                       !Continue program execution
.
.
100 END
110 SUB Errmsg
120   DIM Message$[256]
130   CLEAR 709                                 !When an error occurs, clear the module to
                                                !regain control.
140   B  = SPOLL (70914)                        !Execute a Serial Poll to clear the Service
                                                !Request bit in the Status Register.
150   REPEAT
```

```
160     OUTPUT 70914; "SYST:ERR?"          !Read all error messages in the error queue.
170     ENTER 70914; Code, Message$
180     PRINT Code, Message$
190   UNTIL Code = 0
200   OUTPUT 70914; "*CLS"                 !Clear all bits in the module Standard Event
                                           Status Register
210   STOP
220 SUBEND
```

## Synchronizing Relay Matrix Switches

You can use the *OPC? common command to synchronize a Relay Matrix Switch module to external measurement instruments.

### Example: Synchronizing a Relay Matrix Switch

This example shows one way to synchronize a Relay Matrix Switch module with measurement instruments. In this example, the module switches a signal to a multimeter. The program then verifies that the channel is closed before the multimeter begins its measurement.

```
10   OUTPUT 70914; "*RST"                 !Reset the module
20   OUTPUT 70914; "CLOS (@10012)"        !Close a channel
30   OUTPUT 70914; "*OPC?"                !Wait for operation complete
40   ENTER 70914; Opc_value
50   OUTPUT 70914; "CLOS? (@10012)"       !Test that the channel is closed
60   ENTER 70914; A
70   OUTPUT 70903; "MEAS:VOLT:DC?"        !When channel is closed, measure
                                          !the voltage
80   ENTER 70903; Meas_value
90   PRINT Meas_value                     !Print the measured value
100 END
```

# Chapter 3
# Relay Matrix Switch Command Reference

## About This Chapter

This chapter describes the Standard Commands for Programmable Instruments (SCPI) and the IEEE 488.2 Common commands for the E1468A and E1469A Relay Matrix Switch modules. See the appropriate command module user's manual for additional information on SCPI and Common commands. This chapter contains the following sections:


- Command Types. . . . . .37
- SCPI Command Reference . . . . . .40
- IEEE 488.2 Common Commands Quick Reference. . . . . .65
- SCPI Commands Quick Reference . . . . . .66


## Command Types

Commands are separated into two types: IEEE 488.2 Common commands and SCPI commands.

### Common Command Format

The IEEE 488.2 standard defines the Common commands that perform functions like reset, self-test, status byte query, etc. Common commands are four or five characters in length, always begin with an asterisk (*), and may include one or more parameters. The command keyword is separated from the first parameter by a space character. Some examples of Common commands are:

*RST, *ESE <*mask*>, *STB?

### SCPI Command Format

SCPI commands perform functions like closing switches, making measurements, and querying instrument states or retrieving data. A subsystem command structure is a hierarchical structure that usually consists of a top-level (or root) command, one or more lower-level commands, and their parameters. The following example shows part of a typical subsystem:

```
[ROUTe:]
    CLOSe <channel_list>
    SCAN <channel_list>
        MODE?
```

[ROUTe:] is the optional root command, CLOSe and SCAN are second-level commands with parameters, and :MODE? is a third-level command. [ROUTe:] is an implied command and is, therefore, optional.

## Command Separator

A colon (:) always separates one command from the next lower-level command, such as [ROUTe:]SCAN:MODE? Colons separate the root command from the second-level command ([ROUTe:]SCAN) and the second level from the third level (SCAN:MODE?).

## Abbreviated Commands

The command syntax shows most commands as a mixture of upper- and lowercase letters. The uppercase letters indicate the abbreviated spelling for the command. For shorter program lines, send the abbreviated form. For better program readability, you may send the entire command. The instrument will accept either the abbreviated form or the entire command.

For example, if the command syntax shows DIAGnostic, DIAG and DIAGNOSTIC are both acceptable forms. Other forms of DIAGnostic, such as DIAGN or DIAGNOS will generate an error. You may use upper- or lowercase letters. Therefore, DIAGNOSTIC, diagnostic, and DiAgNoStIc are all acceptable.

## Implied Commands

Implied commands appear in square brackets ([ ]) in the command syntax. *The brackets are not part of the command and are not sent to the instrument.* Suppose you send a second-level command but do not send the preceding implied command. In this case, the instrument assumes you intended to use the implied command and it responds as if you had sent it. Examine the [SOURce] subsystem shown below:

```
[SOURce:]
    PULSe
    :COUNt
    :COUNt?
    :PERiod
    :PERiod?
```

The root command [SOURce:] is an implied command. To set the instrument's pulse count to 25, you can send either of the following command statements:

```
SOUR:PULS:COUN 25 or PULS:COUN 25
```

## Variable Command Syntax

Some commands have what appears to be a variable syntax. For example, OUTP:ECLTn and OUTP:TTLTn. In these commands, the n is replaced by a number. No space is left between the command and the number because the number is not a parameter. The number is part of the command syntax. In the case of OUTP:ECLT*n*, *n* can range from 0 to 1. In OUTP:TTLT*n*, *n* can range from 0 through 7.

### Parameter Types

The following table contains explanations and examples of parameter types you may see in this chapter.

| Type | Explanations and Examples |
|---|---|
| Boolean | Boolean parameters represent a single binary condition that is either true or false (ON, OFF, 1, 0). Any non-zero value is considered true. |
| Discrete | Discrete parameters selects from a finite number of values. These parameters use mnemonics to represent each valid setting. An example is TRIGger:SOURce <*source*>, where *source* can be BUS, EXTernal, HOLD, IMMediate, ECLTrgn, or TTLTrgn. |
| Numeric | Numeric Parameters are commonly used decimal representations of numbers including optional signs, decimal points, and scientific notation (for example, 123, 123E2, -123, -1.23E2, .123, 1.23E-2, 1.23000E- 01). Special cases include MIN, MAX, DEFault, and INFinity. |
| Optional | Optional Parameters are shown within square brackets ([]). The brackets are not part of the command and are not sent to the instrument. If you do not specify a value for an optional parameter, the instrument chooses a default value.<br><br>For example, consider ARM:COUNt?[MIN\|MAX]. If you send the command without specifying a parameter, the present ARM:COUNt value is returned. If you send the MIN parameter, the command returns the minimum count available. If you send the MAX parameter, the command returns the maximum count available. Be sure to place a space between the command and the parameter. |

## Linking Commands

**Linking IEEE 488.2 Common Commands with SCPI Commands.** Use a semicolon (;) between the commands. For example, *RST;OUTP ON or TRIG:SOUR HOLD;*TRG.

**Linking Multiple SCPI commands.** Use both a semicolon (;) and a colon (:) between the commands, such as ARM:COUN 1;:TRIG:SOUR EXT.

# SCPI Commands Reference

This section describes the Standard Commands for Programmable Instruments (SCPI) commands for the Relay Matrix Switch modules. Commands are listed alphabetically by subsystem and within each subsystem.

# ABORt

The ABORt command subsystem stops a scan in progress when the scan is enabled via the interface and the trigger source is TRIGger:SOURce BUS or TRIGger:SOURce HOLD.

## Subsystem Syntax

ABORt

## Comments

**ABORt Actions:** ABORt stops the scan and invalidates the current *channel_list*.

**Stopping a Scan Enabled Via Interface:** When a scan is enabled via an interface, an interface CLEAR command (CLEAR 7) can be used to stop the scan. When the scan is enabled via the interface and TRIG:SOUR BUS or HOLD is set, you can use ABORt to stop the scan.

**Restarting a Scan:** Use the INIT command to restart the scan.

**Related Commands:** ARM, INITiate:CONTinuous, [ROUTe:]SCAN, TRIGger

## Example

**Stopping a Scan with ABORt**

This example stops a (continuous) scan in progress.

```
TRIG:SOUR BUS                 ! *TRG command is trigger source
INIT:CONT ON                  ! Set continuous scanning
SCAN (@10000:10003)           ! Scan channels 00-03
INIT                          ! Start scan, close channel 00
.
.
ABOR                          ! Abort scan in progress.
```

# ARM

The ARM subsystem selects the number of scanning cycles (1 to 32,767) for each INITiate command.

### Subsystem Syntax

```
ARM
  :COUNt <number> MIN | MAX
  :COUNt? [MIN | MAX]
```

## ARM:COUNt

**ARM:COUNt <*number*> MIN | MAX** allows scanning cycles to occur a multiple of times (1 to 32,767) with one INITiate command when INITiate:CONTinuous OFF | 0 is set. MIN sets 1 cycle and MAX sets 32,767 cycles.

### Parameters

| Name | Type | Range of Values | Default Value |
|---|---|---|---|
| <*number*> | numeric | 1-32,767 \| MIN \| MAX | 1 |

### Comments

**Number of Scans:** Use only values between 1 and 32,767 for the number of scanning cycles.

**Related Commands:** ABORt, INITiate[:IMMediate]

***RST Condition:** ARM:COUNt 1

### Example

**Setting Ten Scanning Cycles**

This example sets a Relay Matrix Switch module for 10 scans of channels 00 through 03.

```
ARM:COUN 10                 !Set 10 scans per INIT command
SCAN (@10000:10003)         !Scan channels 00-03
INIT                        !Start scan, close channel 00
```

# ARM:COUNt?

**ARM:COUNt? [MIN | MAX]** returns the current number of scanning cycles set by ARM:COUNt. The current number of scan cycles is returned when MIN or MAX is not supplied. With MIN or MAX as a parameter, MIN returns 1 and MAX returns 32767.

## Parameters

| Name | Type | Range of Values | Default Value |
|---|---|---|---|
| MIN \| MAX | numeric | MIN = 1, MAX = 32,767 | current cycles |

## Comments

**Related Command:** INITiate[:IMMediate]

## Example

**Query Number of Scans**

This example sets a switchbox for 10 scanning cycles and queries the number of scan cycles set. The ARM:COUN? command returns 10.

```
ARM:COUN 10          !Set 10 scans per INIT command
ARM:COUN?            !Query number of scans
```

# INITiate

The INITiate command subsystem selects continuous scanning cycles and starts the scanning cycle.

### Subsystem Syntax

```
INITiate
  :CONTinuous <mode>
  :CONTinuous?
  [:IMMediate]
```

## INITiate:CONTinuous

**INITiate:CONTinuous <*mode*>** enables or disables continuous scanning cycles for the switchbox.

### Parameters

| Name | Type | Range of Values | Default Value |
|---|---|---|---|
| <*mode*> | boolean | 0 \| 1 \| OFF \| ON | 0 \| OFF |

### Comments

**Continuous Scanning Operation:** Continuous scanning is enabled with the INITiate:CONTinuous ON or INITiate:CONTinuous 1 command. Sending the INITiate[:IMMediate] command closes the first channel in the channel list. Each trigger from the source specified by the TRIGger:SOURce command advances the scan through the channel list. A trigger at the end of the channel list closes the first channel in the channel list and the scan cycle repeats.

**Non-Continuous Scanning Operation:** Non-continuous scanning is enabled with the INITiate:CONTinuous OFF or INITiate:CONTinuous 0 command. Sending the INITiate[:IMMediate] command closes the first channel in the channel list. Each trigger from the source specified by the TRIGger:SOURce command advances the scan through the channel list. At the end of the scanning cycle, the last channel in the channel list is closed and the scanning cycle stops.

**Stopping Continuous Scan:** See the ABORt command.

**Related Commands:** ABORt, ARM:COUNt, TRIGger:SOURce

***RST Condition:** INITiate:CONTinuous OFF | 0

### Example

**Enabling Continuous Scanning**

This example enables continuous scanning of channels 00 through 03 of a single-module switchbox. Since TRIGger:SOURce IMMediate (default) is set, use an interface clear command (such as CLEAR) to stop the scan.

```
INIT:CONT ON                    !Enable continuous scanning
SCAN (@10000:10003)             !Scan channels 00-03
INIT                            !Start scan cycle, close chan 00
```

## INITiate:CONTinuous?

**INITiate:CONTinuous?** queries the scanning state. With continuous scanning enabled, the command returns 1. With continuous scanning disabled, the command returns 0.

### Example

**Query Continuous Scanning State**

This example enables continuous scanning of a switchbox and queries the state. Since continuous scanning is enabled, INIT:CONT? returns 1.

```
INIT:CONT ON                    !Enable continuous scanning
INIT:CONT?                      !Query continuous scanning state
```

## INITiate[:IMMediate]

**INITiate[:IMMediate]** starts the scanning process and closes the first channel in the channel list. Successive triggers from the source selected by the TRIGger:SOURce command advance the scan through the channel list.

### Comments

**Starting the Scanning Cycle:** The INITiate[:IMMediate] command starts scanning by closing the first channel in the channel list. Each trigger received advances the scan to the next channel in the channel list. An invalid channel list definition causes an error (see [ROUTe:]SCAN).

**Stopping Scanning Cycles:** See ABORt.

### Example

**Enabling a Single Scan**

This example enables a single scan of channels 00 through 03 of a single-module switchbox. The trigger source to advance the scan is immediate (internal) triggering set with TRIGger:SOURce:IMMediate.

```
SCAN (@10000:10003)             !Scan channels 00-03
INIT                            !Begin scan, close channel 00
```

# OUTPut

The OUTPut subsystem selects the source of the output trigger generated when a channel is closed during a scan. The selected output can be enabled, disabled, and queried. The three available outputs are the ECLTrg and TTLTrg trigger buses and the E1406 Command Module front panel Trig Out port.

### Subsystem Syntax

```
OUTPut
    :ECLTrgn (:ECLTrg0 or :ECLTrg1)
        [:STATe] <mode>
        [:STATe]?
    [:EXTernal]
        [:STATe] <mode>
        [:STATe]?
    :TTLTrgn (:TTLTrg0 through :TTLTrg7)
        [:STATe] <mode>
        [:STATe]?
```

## OUTPut:ECLTrg[:STATe]

**OUTPut:ECLTrg*n*[:STATe] <*mode*>** selects and enables which ECL Trigger bus line (0 or 1) will output a trigger when a channel is closed during a scan. This is also used to disable a selected ECL Trigger bus line. *n* specifies the ECL Trigger bus line (0 or 1) and *mode* enables (ON or 1) or disables (OFF or 0) the specified ECLTrg bus line.

### Parameters

| Name | Type | Range of Values | Default Value |
|---|---|---|---|
| *n* | numeric | 0 or 1 | N/A |
| <*mode*> | boolean | 0 \| 1 \| OFF \| ON | 0 \| OFF |

### Comments

**Enabling ECL Trigger Bus:** When enabled, a pulse is output from the selected ECL Trigger bus line (0 or 1) after each channel is closed during a scan. If disabled, a pulse is not output. The output is a negative-going pulse.

**ECL Trigger Bus Line Shared by Switchboxes:** Only one switchbox configuration can use the selected trigger at a time. When enabled, the selected ECL Trigger bus line (0 or 1) is pulsed by the switchbox each time a scanned channel is closed. To disable the output for a specific switchbox, send the OUTPut:ECLTrg*n* OFF or 0 command for that switchbox.

**One Output Selected at a Time:** Only one output (ECLTrg 0 or 1; TTLTrg 0, 1, 2, 3, 4, 5, 6, or 7; or EXTernal) can be enabled at one time. Enabling a different output source will automatically disable the active output. For example, if TTLTrg1 is the active output, and TTLTrg4 is enabled, TTLTrg1 will become disabled and TTLTrg4 will become the active output.

**Related Commands:** [ROUTe:]SCAN, TRIGger:SOURce, OUTPut:ECLTrg[:STATe]?

***RST Condition:** OUTPut:ECLTrg[:STATe] OFF (disabled).

### Example

**Enabling ECL Trigger Bus Line 0**

```
OUTP:ECLT0:STAT 1          ! Enable ECL Trigger bus line 0 to
                           !output pulse after each scanned
                           !channel is closed.
```

## OUTPut:ECLTrg[:STATe]?

**OUTPut:ECLTrg[:STATe]?** queries the present state of the specified ECL Trigger bus line. The command returns 1 if the specified ECLTrg bus line is enabled or 0 if disabled.

### Example

**Query ECL Trigger Bus Enable State**

This example enables ECL Trigger bus line 0 and queries the enable state. The OUTPut:ECLTrg*n*? command returns 1 since the port is enabled.

```
OUTP:ECLT0:STAT 1          ! Enable ECL Trigger bus line 0
OUTP:ECLT0?                ! Query bus enable state
```

## OUTPut[:EXTernal][:STATe]

**OUTPut[:EXTernal][:STATe] <*mode*>** enables or disables the Trig Out port on the E1406 Command Module to output a trigger when a channel is closed during a scan. ON | 1 enables the port and OFF | 0 disables the port.

### Parameters

| Name | Type | Range of Values | Default Value |
|---|---|---|---|
| <*mode*> | boolean | 0 \| 1 \| OFF \| ON | 0 \| OFF |

### Comments

**Enabling Trig Out Port:** When enabled, a pulse is output from the Trig Out port after each scanned switchbox channel is closed. If disabled, a pulse is not output from the port after channel closures. The output is a negative going pulse.

**Trig Out Port Shared by Switchboxes:** Only one switchbox configuration can use the selected trigger at a time. When enabled, the Trig Out port is pulsed by the switchbox each time a scanned channel is closed. To disable the output for a specific switchbox, send the OUTP OFF or 0 command for that switchbox.

**One Output Selected at a Time:** Only one output (ECLTrg 0 or 1; TTLTrg 0, 1, 2, 3, 4, 5, 6, or 7; or EXTernal) can be enabled at one time. Enabling a different output source will automatically disable the active output. For example, if TTLTrg1 is the active output, and TTLTrg4 is enabled, TTLTrg1 will become disabled and TTLTrg4 will become the active output.

**Related Commands:** [ROUTe:]SCAN, TRIGger:SOURce, OUTPut[:EXTernal][:STATe]?

***RST Condition:** OUTPut[:EXTernal][:STATe] OFF (disabled).

### Example

**Enabling Trig Out Port**

```
OUTP:EXT 1          !Enable Trig Out port to output
                    !pulse after each scanned channel
                    !is closed
```

## OUTPut[:EXTernal][:STATe]?

**OUTPut[:EXTernal][:STATe]?** queries the present state of the Trig Out port. The command returns 1 if the port is enabled or 0 if disabled.

### Example

**Query Trig Out Port Enable State**

This example enables the Trig Out port and queries the enable state. The OUTPut? command returns 1 since the port is enabled.

```
OUTP:EXT ON         !Enable Trig Out port
OUTP:EXT?           !Query port enable state
```

# OUTPut:TTLTrg[:STATe]

**OUTPut:TTLTrg*n*[:STATe] <*mode*>** selects and enables which TTL Trigger bus line (0 to 7) will output a trigger when a channel is closed during a scan. This is also used to disable a selected TTL Trigger bus line. *n* specifies the TTL Trigger bus line (0 to 7) and *mode* enables (ON or 1) or disables (OFF or 0) the specified TTL Trigger bus line.

## Parameters

| Name | Type | Range of Values | Default Value |
|---|---|---|---|
| *n* | numeric | 0 or 1 | N/A |
| <*mode*> | boolean | 0 \| 1 \| OFF \| ON | 0 \| OFF |

## Comments

**Enabling TTL Trigger Bus:** When enabled, a pulse is output from the selected TTL Trigger bus line (0 to 7) after each channel in the switchbox is closed during a scan. If disabled, a pulse is not output. The output is a negative-going pulse.

**TTL Trigger Bus Line Shared by Switchboxes:** Only one switchbox configuration can use the selected TTL Trigger at a time. When enabled, the selected TTL Trigger bus line (0 to 7) is pulsed by the switchbox each time a scanned channel is closed. To disable the output for a specific switchbox, send the OUTPut:TTLTrg*n* OFF or 0 command for that switchbox.

**One Output Selected at a Time:** Only one output (ECLTrg 0 or 1; TTLTrg 0, 1, 2, 3, 4, 5, 6, or 7; or EXTernal) can be enabled at one time. Enabling a different output source will automatically disable the active output. For example, if TTLTrg1 is the active output, and TTLTrg4 is enabled, TTLTrg1 will become disabled and TTLTrg4 will become the active output.

**Related Commands:** [ROUTe:]SCAN, TRIGger:SOURce, OUTPut:TTLTrg[:STATe]?

***RST Condition:** OUTPut:TTLTrg[:STATe] OFF (disabled).

## Example

**Enabling TTL Trigger Bus Line 7**

```
OUTP:TTLT7:STAT 1          ! Enable TTL Trigger bus line 7 to
                           !output pulse after each scanned
                           !channel is closed
```

# OUTPut:TTLTrg[:STATe]?

**OUTPut:TTLTrg[:STATe]?** queries the present state of the specified TTL Trigger bus line. The command returns 1 if the specified TTLTrg bus line is enabled or 0 if disabled.

## Example

**Query TTL Trigger Bus Enable State**

This example enables TTL Trigger bus line 7 and queries the enable state. The OUTPut:TTLTrg*n*? command returns 1 since the port is enabled.

```
OUTP:TTLT7:STAT 1          !Enable TTL Trigger bus line 7
OUTP:TTLT7?                !Query bus enable state
```

# [ROUTe:]

The [ROUTe:] subsystem controls switching and scanning operations for Relay Matrix Switch modules in a switchbox.

**NOTE** *The [ROUTe:] subsystem opens all previously closed relays. Therefore, it should be the first relay configuration command.*

### Subsystem Syntax

[ROUTe:]
CLOSe <*channel_list*>
CLOSe? <*channel_list*>
OPEN <*channel_list*>
OPEN? <*channel_list*>
SCAN <*channel_list*>

## [ROUTe:]CLOSe

**[ROUTe:]CLOSe <*channel_list*>** closes the Relay Matrix Switch channels specified by channel_list.

### Parameters

| Name | Type | Range of Values | Default Value |
|---|---|---|---|
| <*channel_list*> | numeric | E1468A: r = 0 to 7<br>c = 0 to 7<br>E1469A: rr = 00 to 03<br>cc =00 to 15 | N/A |

### Comments

***channel_list* Form:** For the E1468A, *channel_list* has the form (@*ssrc*) where *ss* = card number (01-99), *r* = row number, and *c* = column number. For the E1469A, *channel_list* has the form (@*ssrrcc*) where *ss* = card number (01-99), *rr* = row number, and *cc* = column number.

**Closing Channels (E1468A Only):**

- For a single channel, use [ROUT:]CLOS (@*ssrc*)
- For multiple channels, use [ROUT:]CLOS (@*ssrc,ssrc*,...)
- For sequential channels, use [ROUT:]CLOS (@*ssrc:ssrc*)
- for groups of sequential channels use [ROUT:]CLOS (@*ssrc:ssrc,ssrc:ssrc*).

You can use any combination of these commands. However, closure order for multiple channels with a single command is not guaranteed.

**Closing Channels (E1469A Only):**

- For a single channel, use [ROUT:]CLOS (@*ssrrcc*)
- For multiple channels, use [ROUT:]CLOS (@*ssrrcc,ssrrcc*,...)
- For sequential channels, use [ROUT:]CLOS (@*ssrrcc:ssrrcc*)
- for groups of sequential channels use [ROUT:]CLOS (@*ssrrcc:ssrrcc,ssrrcc:ssrrcc*).

You can use any combination of these commands. However, closure order for multiple channels with a single command is not guaranteed.

**Related Commands:** [ROUTe:]OPEN, [ROUTe:]CLOSe?

***RST Condition:** All channels open.

### Example

**Closing Relay Matrix Switch Module Channels**

This example closes channels 10100 and 20013 of a two-module switchbox (card numbers 01 and 02).

```
CLOS (@10100,20013)          !Close channels 10100 and
                             !20013. 10100 closes row 01,
                             !column 00 of card #1 and 20013
                             !closes row 00, column 13 on
                             !card #2.
```

## [ROUTe:]CLOSe?

**[ROUTe:]CLOSe? <*channel_list*>** returns the current state of the channel(s) queried. *channel_list* has the form (@*ssrc*) or (@*ssrrcc*) (see [ROUTe:]CLOSe for definition). The command returns 1 if channel(s) are closed or returns 0 if channel(s) are open.

### Comments

**Query is Software Readback:** The [ROUTe:]CLOSe? command returns the current software state of the channel(s) specified. It does not account for relay hardware failures. A maximum of 127 channels at a time can be queried for a multi-module switchbox.

### Example

**Query Channel Closures**

This example closes channels 10100 and 20013 of a two-module switchbox and queries channel closure. Since the channels are programmed to be closed, 1, 1 is returned as a string.

```
CLOS (@10100,20013)          !Close channels 10100 and
                             !20013. 10100 closes row 01,
                             !column 00 of card #1 and 20013
                             !closes row 00, column 13 on
                             !card #2.
CLOS? (@10100,20013)         !Query channel closures
```

# [ROUTe:]OPEN

**[ROUTe:]OPEN <*channel_list*>** opens the Relay Matrix Switch channels specified by *channel_list*.

## Parameters

| Name | Type | Range of Values | Default Value |
|---|---|---|---|
| <*channel_list*> | numeric | E1468A: r = 0 to 7<br>c = 0 to 7<br>E1469A: rr = 00 to 03<br>cc =00 to 15 | N/A |

## Comments

***channel_list* Form:** For the E1468A, *channel_list* has the form (@*ssrc*) where *ss* = card number (01-99), *r* = row number, and *c* = column number. For the E1469A, *channel_list* has the form (@*ssrrcc*) where *ss* = card number (01-99), *rr* = row number, and *cc* = column number.

**Opening Channels (E1468A Only):**

- For a single channel, use [ROUT:]OPEN (@*ssrc*)
- For multiple channels, use [ROUT:]OPEN (@*ssrc,ssrc*,...)
- For sequential channels, use [ROUT:]OPEN (@*ssrc:ssrc*)
- for groups of sequential channels use [ROUT:]OPEN (@*ssrc:ssrc,ssrc:ssrc*).

You can use any combination of these commands. However, closure order for multiple channels with a single command is not guaranteed.

**Opening Channels (E1469A Only):**

- For a single channel, use [ROUT:]OPEN (@*ssrrcc*)
- For multiple channels, use [ROUT:]OPEN (@*ssrrcc,ssrrcc*,...)
- For sequential channels, use [ROUT:]OPEN (@*ssrrcc:ssrrcc*)
- for groups of sequential channels use [ROUT:]OPEN (@*ssrrcc:ssrrcc,ssrrcc:ssrrcc*).

You can use any combination of these commands. However, closure order for multiple channels with a single command is not guaranteed.

**Related Commands:** [ROUTe:]CLOSe, [ROUTe:]OPEN?

***RST Condition:** All channels open.

## Example

**Opening Channels**

This example opens channels 10100 and 20013 of a two-module switchbox (card numbers 01 and 02).

```
OPEN (@10100,20013)          !Open channels 10100 and 20013
```

# [ROUTe:]OPEN?

**[ROUTe:]OPEN? <*channel_list*>** returns the current state of the channel(s) queried. *channel_list* has the form (@*ssrc*) or (@*ssrrcc*) (see [ROUTe:]OPEN for definition). The command returns 1 if channel(s) are open or returns 0 if channel(s) are closed.

## Comments

**Query is Software Readback:** The [ROUTe:]OPEN? command returns the current software state of the channels specified. It does not account for relay hardware failures. A maximum of 127 channels at a time can be queried for a multi-module switchbox.

## Example

**Query Channel Open State**

This example opens channels 10100 and 20013 of a two-module switchbox and queries channel 20013 state. Since channel 20013 is programmed to be open, `1` is returned.

```
OPEN (@10100,20013)        !Open channels 10100 and 20013
OPEN? (@20013)             !Query channel open state
```

# [ROUTe:]SCAN

**[ROUTe:]SCAN <*channel_list*>** defines the channels to be scanned.

## Parameters

| Name | Type | Range of Values | Default Value |
|---|---|---|---|
| <*channel_list*> | numeric | E1468A: r = 0 to 7<br>c = 0 to 7<br>E1469A: rr = 00 to 03<br>cc =00 to 15 | N/A |

## Comments

***channel_list* Form:** For the E1468A, *channel_list* has the form (@*ssrc*) where *ss* = card number (01-99), *r* = row number, and *c* = column number. For the E1469A, *channel_list* has the form (@*ssrrcc*) where *ss* = card number (01-99), *rr* = row number, and *cc* = column number.

**Defining Scan List:** When [ROUTe:]SCAN is executed, the *channel_list* is checked for valid card and channel numbers. An error is generated for an invalid *channel_list*.

**Scanning Channels (E1468A Only):**

- For a single channel, use [ROUT:]SCAN (@*ssrc*)
- For multiple channels, use [ROUT:]SCAN (@*ssrc,ssrc*,...)
- For sequential channels, use [ROUT:]SCAN (@*ssrc:ssrc*)
- for groups of sequential channels use [ROUT:]SCAN (@*ssrc:ssrc,ssrc:ssrc*).

You can use any combination of these commands. However, closure order for multiple channels with a single command is not guaranteed.

**Scanning Channels (E1469A Only):**

- For a single channel, use [ROUT:]SCAN (@*ssrrcc*)
- For multiple channels, use [ROUT:]SCAN (@*ssrrcc,ssrrcc*,...)
- For sequential channels, use [ROUT:]SCAN (@*ssrrcc:ssrrcc*)
- for groups of sequential channels use [ROUT:]SCAN (@*ssrrcc:ssrrcc,ssrrcc:ssrrcc*).

You can use any combination of these commands. However, closure order for multiple channels with a single command is not guaranteed.

**Scanning Operation:** When a valid *channel_list* is defined, INITiate[:IMMediate] begins the scan and closes the first channel in the *channel_list*. Successive triggers from the source specified by TRIGger:SOURce advance the scan through the *channel_list*. At the end of the scan, the last trigger opens the last channel.

**Stopping Scan:** See ABORt.

**Related Commands:** TRIGger:SOURce

***RST Condition:** All channels open.

## Example

**Scanning Channels**

This example sets the channels to be scanned from 100 to 200 for a single-module switchbox and initiates the scan sequence.

```
SCAN (@100,200)          !Set scan sequence from ch 100 through 200
INIT                     !Begin scan and close ch 100
```

# STATus

The STATus subsystem reports the bit values of the Operation Status Register (in the command module). It also allows you to unmask the bits you want reported from the Standard Event Register and to read the summary bits from the Status Byte register.

## Subsystem Syntax

```
STATus
    :OPERation
        :CONDition?
        :ENABle <unmask>
        :ENABle?
        [:EVENt?]
    :PRESet
```

The STATus system contains four software registers that reside in a SCPI driver, not in the hardware (see Figure 3-1) Two registers are under IEEE 488.2 control: the Standard Event Status Register (*ESE?) and the Status Byte Register (*STB).

The Operational Status bit (OPR), Service Request bit (RSQ), Standard Event summary bit (ESB), Message Available bit (MAV) and Questionable Data bit (QUE) in the StatusByte Register (bits 7, 6, 5, 4 and 3 respectively) can be queried with the *STB? command.

Use the *ESE? command to query the *unmask* value for the Standard Event Status Register (the bits you want logically OR'd into the summary bit). The registers are queried using decimal weighted bit values. The decimal equivalents for bits 0 through 15 are included in Figure 3-1.

A numeric value of 256 executed in a STATus:OPERation:ENABle <*unmask*> command allows only bit 8 to generate a summary bit. The decimal value for bit 8 is 256.

The decimal values are also used in the inverse manner to determine which bits are set from the total value returned by an EVENt or CONDition query. The SWITCH driver exploits only bit 8 of Operation Status Register. This bit is called the Scan Complete bit which is set whenever a scan operation completes. Since completion of a scan operation is an event in time, bit 8 will never appear set when STAT:OPER:COND? is queried. However, bit 8 is set with the STAT:OPER:EVENt? query command.

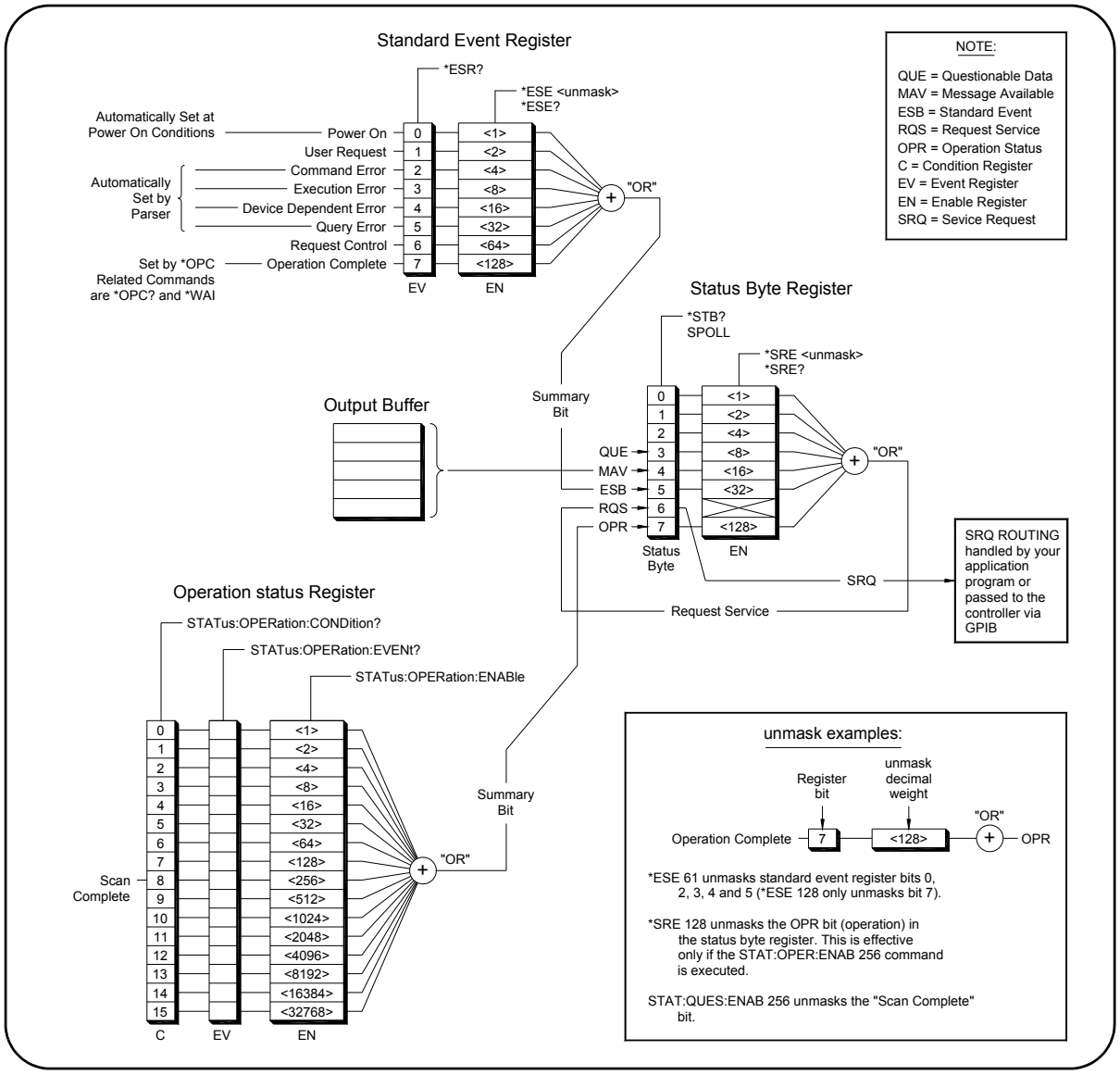



**Figure 3-1. E1468A/E1469A Status System Register Diagram**

# STATus:OPERation:CONDition?

**STATus:OPERation:CONDition?** returns the state of the Condition Register in the Operation Status Group. The state represents conditions which are part of the instrument's operation. The switch module driver does not set bit 8 in this register (see STATus:OPERation[:EVENt]?).

## STATus:OPERation:ENABle

**STATus:OPERation:ENABle <*unmask*>** sets an enable mask to allow events recorded in the Event Register to send a summary bit to the Status Byte Register (bit 7). For Relay Matrix Switch modules, when bit 8 in the Operation Status Register is set to 1 and is enabled by the STAT:OPER:ENABle command, bit 7 in the Status Register is set to 1.

### Parameters

| Name | Type | Range of Values | Default Value |
|---|---|---|---|
| <*unmask*> | numeric | 0 through 65,535 | N/A |

### Comments

**Setting Bit 7 of the Status Byte Register:** STATus:OPERation:ENABle 256 sets bit 7 of the Status Byte Register to 1 after bit 8 of the Operation Status Register is set to 1.

**Related Commands:** [ROUTe:]SCAN

### Example

**Enabling Operation Status Register Bit 8**

```
STAT:OPER:ENAB 256          !Enables bit 8 of the Operation
                            !Status Enable Register to be
                            !reported to bit 7 (OPR) in the
                            !Status Register.
```

## STATus:OPERation:ENABle?

**STATus:OPERation:ENABle?** returns the bit value of the Operation Status Register.

### Comments

**Output Format:** Returns a decimal weighted value from 0 to 65,535 indicating which bits are set to true.

**Maximum Value Returned:** The value returned is the value set by the STAT:OPER:ENAB <*unmask*> command. However, the maximum decimal weighted value used in this module is 256 (bit 8 set to true).

### Example

**Query the Operation Status Enable Register**

```
STAT:OPER:ENAB?             !Queries the Operation Status
                            !Enable Register
```

## STATus:OPERation[:EVENt]?

**STATus:OPERation[:EVENt]?** returns which bits in the Event Register (Operation Status Group) are set. The Event Register indicates when there has been a time-related instrument event.

### Comments

**Setting Bit 8 of the Operation Status Register:** Bit 8 (Scan Complete) is set to 1 after a scanning cycle completes. Bit 8 returns to 0 (zero) after sending the STATus:OPERation[:EVENt]? command.

**Returned Data After Sending STATus:OPERation[:EVENt]?** The command returns +256 if bit 8 of the Operation Status Register is set to 1. The command returns +0 if bit 8 of the Operation Status Register is set to 0.

**Event Register Cleared:** Reading the Event Register with the STATus:OPERation[:EVENt]? command clears it.

**ABORting a Scan:** Aborting a scan will leave bit 8 set to 0.

**Related Commands:** [ROUTe:]SCAN

### Example

**Reading the Operation Status Register After a Scanning Cycle**

| | |
|---|---|
| STAT:OPER? | *!Returns the bit values of the !Operation Status Register.* |
| read the register value | *!+256 shows bit 8 is set to 1. !+0 shows bit 8 is set to 0.* |

## STATus:PRESet

**STATus:PRESet** affects only the Enable Register by setting all Enable Register bits to 0. It does not affect either the "status byte" or the "standard event status". PRESet does not clear any of the Event Registers.

# SYSTem

The SYSTem subsystem returns the error numbers and error messages in the error queue of a switchbox and returns the types and descriptions of modules (cards) in a switchbox.

### Subsystem Syntax

SYSTem
:CDEScription? <*number*>
:CPON <*number*> | ALL
:CTYPe? <*number*>
:ERRor?

## SYSTem:CDEScription?

**SYSTem:CDEScription? <*number*>** returns the description of a selected module (card) in a switchbox.

### Parameters

| Name | Type | Range of Values | Default Value |
|---|---|---|---|
| <*number*> | numeric | 1 through 99 | N/A |

### Comments

**8x8 Relay Matrix Module Description:** SYSTem:CDEScription? <*number*> returns: `8x8 Relay Matrix`

**4x16 Relay Matrix Module Description:** SYST:CDEScription? <*number*> returns: `4x16 Relay Matrix`

### Example

**Reading the Description of a Card #1 Module**

SYST:CDES? 1 *!Returns the description*

## SYSTem:CPON

**SYSTem:CPON <*number*> | ALL**sets the selected module (card) in a switchbox to its power-on state.

### Parameters

| Name | Type | Range of Values | Default Value |
|---|---|---|---|
| <*number*> | numeric | 1 through 99 | N/A |

### Comments

**Matrix Module Power-On State:** The power-on state is all channels (relays) open. *RST opens all channels of all modules in a switchbox, while SYSTem:CPON <*number*> opens the channels in only the module (card) specified in the command.

### Example

**Setting Card #1 Module to Power-On State**

```
SYST:CPON 1                    ! Sets module #1 to power-on
                               !state
```

## SYSTem:CTYPe?

**SYSTem:CTYPe? <*number*>** returns the module (card) type of a selected module in a switchbox.

### Parameters

| Name | Type | Range of Values | Default Value |
|---|---|---|---|
| <*number*> | numeric | 1 through 99 | N/A |

### Comments

**8x8 Relay Matrix Module Model Number:** SYSTem:CTYPe? <*number*> returns `HEWLETT-PACKARD,E1468A,0,A.02.00` , where the 0 after E1468A is the module serial number (always 0) and A.02.00 is an example of the module revision code number.

**4x16 Relay Matrix Switch Module Model Number:** SYSTem:CTYPe? <*number*> returns `HEWLETT-PACKARD,E1469A,0,A.04.00`, where the 0 after E1469A is the module serial number (always 0) and A.04.00 is an example of the module revision code number.

### Example

**Reading the Model Number of a Card #l Module**

```
SYST:CTYP? 1                   !Return the model number
```

## SYSTem:ERRor?

**SYSTem:ERRor?** returns the error numbers and corresponding error messages in the error queue of a switchbox. See Appendix C for a listing of some switchbox error numbers and messages.

### Comments

**Error Numbers/Messages in the Error Queue:** Each error generated by a switchbox stores an error number and corresponding error message in the error queue. The error message can be up to 255 characters long.

**Clearing the Error Queue:** An error number/message is removed from the queue each time the SYSTem:ERRor? command is sent. The errors are cleared first-in, first-out. When the queue is empty, each following SYSTem:ERRor? command returns `0`, "No error". To clear all error numbers/messages in the queue, execute the *CLS command.

**Maximum Error Numbers/Messages in the Error Queue:** The queue holds a maximum of 30 error numbers/messages for each switchbox. If the queue overflows, the last error number/message in the queue is replaced by `-350`, "Too many errors". The least recent error numbers/messages remain in the queue and the most recent are discarded.

## Example

**Querying the Error Queue**

```
SYST:ERR?                          !Query the error queue
```

# TRIGger

The TRIGger subsystem controls the triggering operation of relay matrix modules in a switchbox.

### Subsystem Syntax

```
TRIGger
    [:IMMediate]
    :SOURce <source>
    :SOURce?
```

## TRIGger[:IMMediate]

**TRIGger[:IMMediate]** causes a trigger event to occur when the defined trigger source is TRIGger:SOURce BUS or TRIGger:SOURce HOLD.

### Comments

**Executing the TRIGger[:IMMediate] Command:** A channel list must be defined with [ROUTe:]SCAN<*channel_list*> and an INITiate[:IMMediate] command must be executed before TRIGger[:IMMediate] will execute.

**BUS or HOLD Source Remains:** If selected, the TRIGger:SOURceBUS or TRIGger:SOURceHOLD commands remain in effect after triggering a switchbox with the TRIGger[:IMMediate] command.

**Related Commands:** INITiate, [ROUTe:]SCAN

### Example

**Advancing Scan Using TRIGger Command**

This example scans a single-module switchbox from channel 00 through 03. Since TRIGger:SOURce HOLD is set, the scan is advanced one channel each time TRIGger is executed.

```
TRIG:SOUR HOLD          !Sets trigger source to HOLD
SCAN (@10000:10003)     !Defines channel list
INIT                    !Begin scan, close channel 00
loop statement          !Start count loop
TRIG                    !Advance scan to next channel
increment loop          !Increment loop count
```

# TRIGger:SOURce

**TRIGger:SOURce <*source*>** specifies the trigger *source* to advance the channel list during scanning.

## Parameters

| Source | Type | Description | Default |
|---|---|---|---|
| BUS | discrete | *TRG or GET command | IMM |
| ECLTrg*n* | numeric | ECL Trigger bus line | IMM |
| EXTernal | discrete | Trig In port | IMM |
| HOLD | discrete | Hold Triggering | IMM |
| IMMediate | discrete | Immediate Triggering | IMM |
| TTLTrg*n* | numeric | TTL Trigger bus line <0 - 7> | IMM |

## Comments

**Enabling the Trigger Source:** The TRIGger:SOURce command only selects the trigger source. The INITiate[:IMMediate] command enables the trigger source.

**Using the TRIG Command:** You can use TRIGger[:IMMediate] to advance the scan when TRIGger:SOURceBUS or TRIGger:SOURceHOLD is selected.

**Using External Trigger Inputs:** With TRIGger:SOURceEXTernal selected, only one switchbox at a time can use the external trigger input at the EI406 Trig In port. The trigger input is assigned to the first switchbox that requested the external trigger source (with a TRIGger:SOURceEXTernal command).

**Assigning External Trigger:** A switchbox assigned with TRIGger:SOURceEXTernal remains assigned to that source until the switchbox trigger source is changed to BUS, ECLT, HOLD, IMMediate, or TTLT. When the source is changed, the external trigger source is available to the next switchbox which requests it (with a TRIGger:SOURceEXTernal command). If a switchbox requests an external trigger input already assigned to another switchbox, an error is generated.

**Using Bus Triggers:** To trigger the switchbox with TRIGger:SOURceBUS selected, use the IEEE 488.2 Common command *TRG or the GPIB Group Execute Trigger (GET) command.

**Trig Out Port Shared by Switchboxes:** See the OUTPut command.

**Related Commands:** ABORt, [ROUTe:]SCAN, OUTPut

***RST Condition:** TRIGger:SOURce IMMediate

### Example Scanning Using External Triggers

This example uses external triggering (TRIG:SOUR EXT) to scan channels 00 through 03 switchbox. The trigger source to advance the scan is the input to the Trig In port on the E1406 Command Module. When INIT is executed, the scan is started and channel 00 is closed. Then each trigger received at the Trig In port advances the scan to the next channel.

```
TRIG:SOUR EXT                 !Select external triggering
SCAN (@10000:10003)           !Scan channels 00 through 03
INIT                          !Begin scan, close channel 00
trigger externally            !Advance scan to next channel
```

### Example Scanning Using Bus Triggers

This example uses bus triggering (TRIG:SOUR BUS) to scan channels 00 through 03 of switchbox. The trigger source to advance the scan is the *TRG command (as set with TRIG:SOUR BUS). When INIT is executed, the scan is started and channel 00 is closed. Then, each *TRG command advances the scan to the next channel.

```
TRIG:SOUR BUS                 !Select interface (bus) triggering
SCAN (@10000:10003)           !Scan channels 00 through 03
INIT                          !Start scan, close channel 00
loop statement                !Loop to scan all channels
*TRG                          !Advance scan using bus
                              !triggering
increment loop                !Increment loop count
```

## TRIGger:SOURce?

**TRIGger:SOURce?** returns the current trigger source for the switchbox. Command returns BUS, ECLT, EXT, HOLD, IMM, or TTLT for sources BUS, ECLTrg, EXTernal, HOLD, IMMediate, or TTLTrg, respectively.

### Example Query Trigger Source

This example sets external triggering and queries the trigger source. Since external triggering is set, TRIG:SOUR? returns EXT.

```
TRIG:SOUR EXT                 !Set external trigger source
TRIG:SOUR?                    !Query trigger source
```

# IEEE 488.2 Common Commands Quick Reference

The following table lists the IEEE 488.2 Common (*) commands that apply to the Relay Matrix Switch modules. For more information on Common Commands, see the ANSI/IEEE Standard 488.2-1987.

| Command | Command Description |
|---|---|
| *CLS | Clears all status registers (see STATus:OPERation[:EVENt]?) and clears error queue. |
| *ESE *<unmask>* | Enables Standard Event. |
| *ESE? | Enables Standard Event Query. |
| *ESR? | Standard Event Register Query. |
| *IDN? | Instrument ID Query; returns identification string of the module. |
| *OPC | Operation Complete. |
| *OPC? | Operation Complete Query. |
| *RCL *<n>* | Recalls the instrument state saved by *SAV. You must reconfigure the scan list. |
| *RST | Resets the module. Opens all channels and invalidates current channel list for scanning. Sets ARM:COUN 1, TRIG:SOUR IMM, and INIT:CONT OFF. |
| *SAV *<n>* | Stores the instrument state but does not save the scan list. |
| *SRE *<unmask>* | Service request enable, enables status register bits. |
| *SRE? | Service request enable query. |
| *STB? | Read status byte query. |
| *TRG | Triggers the module to advance the scan when scan is enabled and trigger source is TRIGger:SOURce BUS. |
| *TST? | Self-test. Executes an internal self-test and returns only the first error encountered. Does not return multiple errors. The following is a list of responses you can obtain where "*cc*" is the card number with the leading zero deleted.<br>+0 if self test passes.<br>+cc01 for firmware error.<br>+cc02 for bus error (problem communicating with the module).<br>+cc03 for incorrect ID information read back from the module's ID register.<br>+cc10 if an interrupt was expected but not received.<br>+cc11 if the busy bit was not held for a sufficient amount of time. |
| *WAI | Wait to Complete. |

# SCPI Commands Quick Reference

This table summarizes SCPI commands for the Relay Matrix Switch modules.

| Command | | Description |
|---|---|---|
| ABORt | | Aborts a scan in progress |
| ARM | :COUNt *<number>* MIN \|MAX<br>:COUNt? [MIN\|MAX] | Multiple scans per INIT command<br>Queries number of scans |
| INITiate | :CONTinuous ON \| OFF<br>:CONTinuous?<br>[:IMMediate] | Enables/disables continuous scanning<br>Queries continuous scan state<br>Starts a scanning cycle |
| OUTPut | :ECLTrg*n*[:STATe] ON\|OFF\|1\|0<br>:ECLTrg*n*[:STATe]?<br>[:EXTernal][:STATe] ON\|OFF\|1\|0<br>[:EXTernal][:STATe]?<br>:TTLTrg*n*[:STATe] ON\|OFF\|1\|0<br>:TTLTrg*n*[:STATe]? | Enables/disables the specified ECL trigger line<br>Queries the specified ECL trigger line<br>Enables/disables the Trig Out port on the E1406<br>Queries the external state<br>Enables/disables the specified TTL trigger line<br>Queries the specified TTL trigger line |
| [ROUTe:] | CLOSe *<channel _list>*<br>CLOSe? *<channel _list>*<br>OPEN *<channel_list>*<br>OPEN? *<channel _list>*<br>SCAN *<channel_list>*<br>SCAN:MODE NONE\|VOLT<br>SCAN:MODE? | Closes channel(s)<br>Queries channel(s) closed<br>Opens channel(s)<br>Queries channel(s) opened<br>Defines channels for scanning<br>Sets scan mode (has no effect on Form C operation)<br>Queries the scan mode |
| STATus | :OPERation:CONDition?<br>:OPERation:ENABle<br>:OPERation:ENABle?<br>:OPERation[:EVENt]?<br>:PRESet | Returns contents of the Operation Condition Register<br>Enables events in the Operation Event Register to be reported<br>Returns the mask value set by the :ENABle command<br>Returns the contents of the Operation Event Register<br>Enables Register bits to 0 |
| SYSTem | :CDEScription? *<number>*<br>:CTYPe? *<number>*<br>:CPON *<number>* \|ALL<br>:ERRor? | Returns description of module in a switchbox<br>Returns the module type<br>Opens all channels on specified module(s)<br>Returns error number/message in a switchbox Error Queue |
| TRIGger | [:IMMediate]<br>:SOURce BUS<br>:SOURce EXTernal<br>:SOURce HOLD<br>:SOURce IMMediate<br>:SOURce ECLTrg*n*<br>:SOURce TTLTrg*n*<br>:SOURce? | Causes a trigger to occur<br>Trigger source is *TRG<br>Trigger source is Trig In (on the command module)<br>Holds off triggering<br>Trigger source is the internal triggers<br>Trigger is the VXIbus ECL trigger bus line *n*<br>Trigger is the VXIbus TTL trigger bus line *n*<br>Queries scan trigger source |

# Appendix A
# Relay Matrix Switch Specifications

| | |
|---|---|
| **Input Characteristics** | |
| **Maximum Voltage Terminal to Terminal:**<br>220 Vdc; 250 $V_{rms}$ | **Maximum Voltage Terminal to Chassis:**<br>220 Vdc; 250 $V_{rms}$ |
| **Maximum Current per Channel (non-inductive):**<br>1 Adc or $ac_{rms}$ (Vmax <30 Vdc or $V_{rms}$)<br>0.3 Adc or $ac_{rms}$ (Vmax <220 Vdc or 250 $V_{rms}$) | **Maximum Power per Channel:**<br>40VA |
| **DC Performance** | |
| **Thermal Offset per Channel:**<br><7μV (differential H-L) | **Closed Channel Resistance:**<br><1.5 Ω initially<br><3.5 Ω at end of relay life |
| **Insulation Resistance (between any two points):**<br>$5x10^6$ Ω at 40°C, 95% RH<br>$5x10^8$ Ω at 25°C, 40% RH | |
| **AC Performance** | |
| **Bandwidth (-3dB):**[1]<br>Z(load) = Z(source) = 50 Ω<br>2-Wire mode (4x16): >10 MHz<br>1-Wire mode (1x128): >3 MHz | **Crosstalk Between Channels @10 kHz:**<br>2-Wire mode (4x16): <-90 dB<br>1-Wire mode (1x128): <-60 dB |
| **Open Channel Capacitance**<br>(channel to channel, channel to common):<br>2-Wire mode (4x16): <-90 dB<br>1-Wire mode (1x128): <-60 dB | **Closed Channel Capacitance (Hi-Lo, Lo-Chassis):**<br>650/700 pF |
| **General** | |
| **Module Size / Device Type:**<br>C-size VXIbus, Register based, A16/D16<br>Interrupter (levels 1-7, jumper selectable) | **Power Requirements:**<br>Voltage: +5 V +24 V<br>Peak Module Current (A) 0.10 0.13<br>Dynamic Module Current (A) 0.10 0.02 |
| **Relay Life:**[2]<br>@ No Load: $5x10^6$ Operations<br>@ Full Load: $10^5$ Operations | **Watts/slot:**<br>5.0<br>**Cooling/slot:**<br>0.08 mm H20 @ 0.42 liter/sec |
| **Terminals:**<br>Screw type, maximum wire size 16AWG | **Operating Temperature:** 0° - 55°C<br>**Operating Humidity:** 65% RH, 0° - 40°C<br>**Net Weight (kg):** 1.6 |

1 The -3 dB BW is typically >25 MHz
2 Relays are subject to normal wear-out based on the number of operations.

## *Notes:*

# Appendix B
# Register-Based Programming

## About This Appendix

This appendix contains the information you can use for register-based programming of the E1468A/E1469A Relay Matrix Switch modules. The contents include:


- Register Addressing . . . . . .69
- Reading the Registers . . . . . .72
- Writing to the Registers . . . . . .73


## Register Addressing

The E1468A/E1469A Relay Matrix Switch modules are register-based modules that do not support the VXIbus word serial protocol. When a SCPI command is sent to the modules, the instrument driver resident in the command module parses the command and programs the module at the register level.

### Addressing Overview

Register-based programming is a series of *reads* and *writes* directly to the module registers. This can increase throughput speed since it eliminates command parsing and allows the use of an embedded controller. It also allows use of an alternate VXI controller, eliminating the command module.

To access a specific register for either read or write operations, the address of the register must be used. Register addresses for the plug-in modules are in an address space known as VXI A16. The exact location of A16 within a VXIbus master's memory map depends on the design of the VXIbus master you are using. For the E1406 Command Module, the A16 space location starts at $1F0000_{16}$.

The A16 space is further divided so that the modules are addressed only at locations above $1FC000_{16}$ within A16. Every module is allocated 64 register addresses ($40_{16}$). The address of a module is determined by its logical address (set by the address switches on the module) times 64 ($40_{16}$). For the E1468A/E1469A modules, the factory setting is 112 ($70_{16}$), so the addresses start at $1C00_{16}$.

Register addresses for register-based devices are located in the upper 25% of VXI A16 address space. Every VXI device (up to 256) is allocated a 64 byte block of addresses. Figure B-1 shows the register address location within A16. Figure B-2 shows the location of A16 address space in the E1406 Command Module.

## The Base Address

When you are reading or writing to a module register, a hexadecimal or decimal register address is specified. This address consists of a base address plus a register offset. The base address used in register-based programming depends on whether the A16 address space is outside or inside the E1406 Command Module.

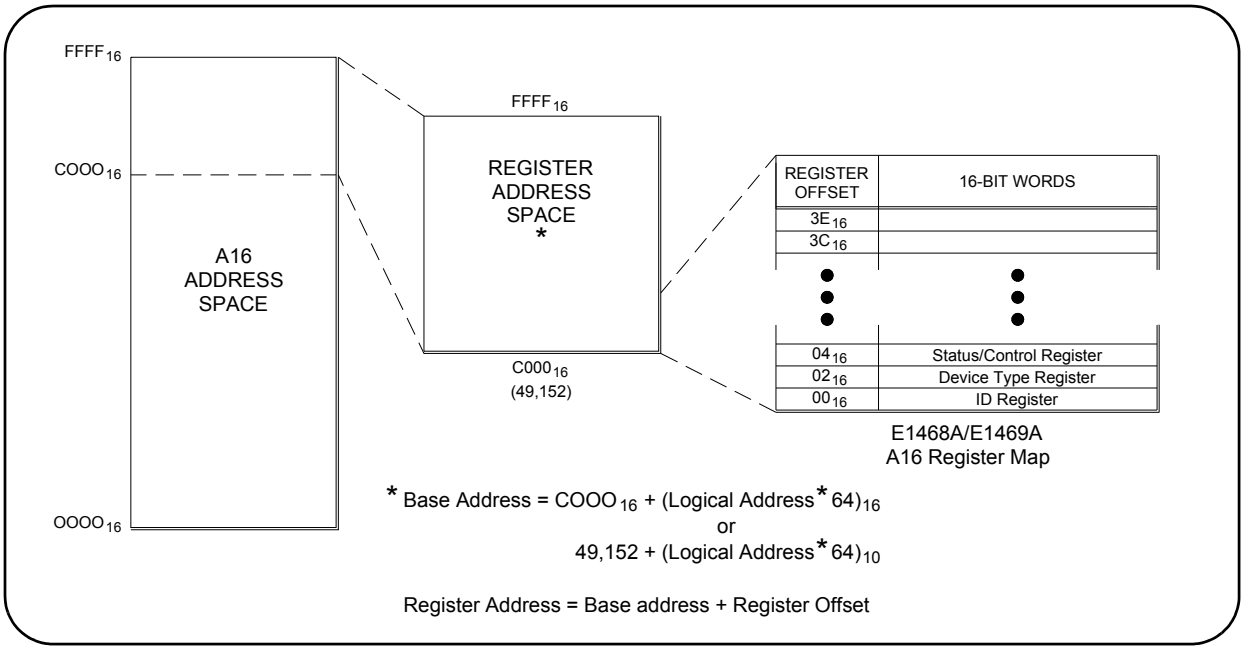



**Figure B-1. Register Address Locations Within VXI A16**

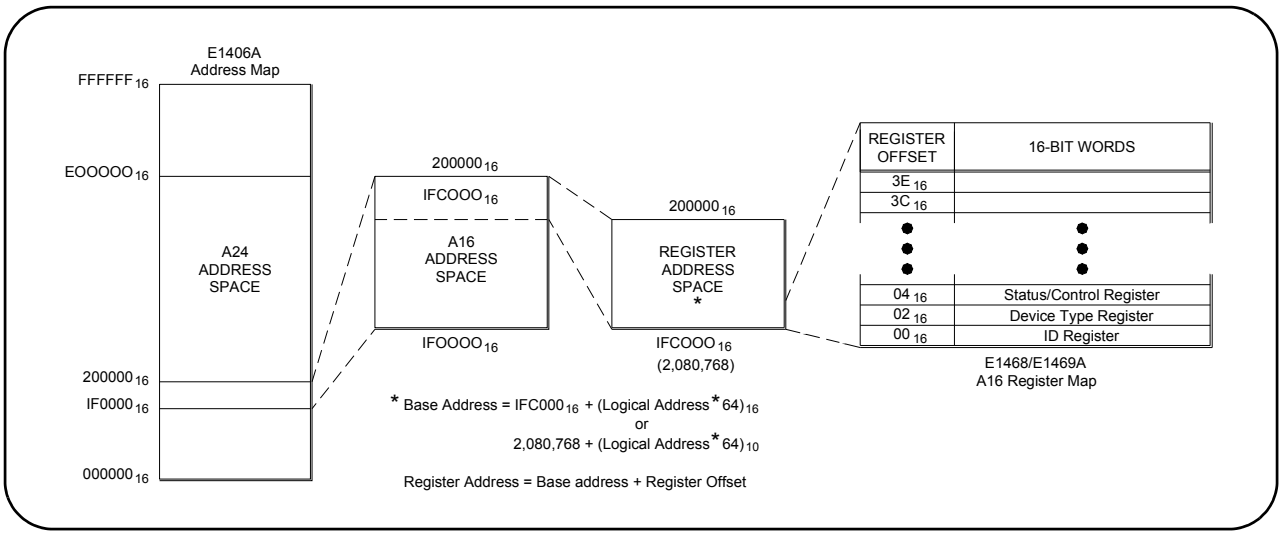



**Figure B-2. A16 Address Space in the E1406 Command Module**

### A16 Address Space Outside the Command Module

When the E1406 Command Module is not part of your VXIbus system, the E1468A/E1469A base address is computed as:

$A16_{base} = 1FC000_{16} + (LADDR_{16} * 64_{16})$

or (decimal)

$A16_{base} = 2{,}080{,}768 + (LADDR * 64)$

where $1FC000_{16}$ (2,080,768) is the starting location of the register addresses, LADDR is the module's logical address, and 64 is the number of address bytes per VXI device.

For example, a Relay Matrix Switch module's Status/Control Register has an offset of $04_{16}$. When you write to or read from this register, the offset is added to the base address to form the register address (using a logical address of 112):

register address = base address + register offset

$= 1FC000_{16} + (112 * 64)_{16} + 04_{16}$

$= 1FC000_{16} + 1C00_{16} + 04_{16} = 1FDC04_{\mathbf{16}}$

*or*

$= 2{,}080{,}768 + (112 * 64) + 4$

$= 2{,}080{,}768 + 7168 + 4 = 2{,}087{,}940$

### A16 Address Space Inside the Command Module or Mainframe

When the A16 address space is inside the E1406 Command Module, the E1468A/E1469A base address is computed as:

$1FC000_{16} + (LADDR_{16} * 64_{16})$

or (decimal)

$2{,}080{,}768 + (LADDR * 64)$

where $1FC000_h$ (2,080,768) is the starting location of the VXI A16 addresses, LADDR is the module's logical address, and 64 is the number of address bytes per register-based device. The E1468A/E1469A factory-set logical address is 112. If this address is not changed, the module will have a base address of:

$1FC000_{16} + (70_{16} * 40_{16}) = 1FC000_{16} + 1C00_{16} = 1FDC00_{16}$

or (decimal)

$2{,}080{,}768 + (112 * 64) = 2{,}080{,}768 + 7168 = 2{,}087{,}936$

## Register Definitions

You can program the E1468A/E1469A modules using their hardware registers. The procedures for reading or writing to a register depend on your operating system and programming language. Whatever the access method, you will need to identify each register with its address.

**E1468A/E1469A Register Map**

| Register Name | Type | Address |
|---|---|---|
| Manufacturer ID | Read Only | base + $00_{16}$ |
| Device Type | Read Only | base + $02_{16}$ |
| Status/Control | Read/Write | base + $04_{16}$ |
| Bank 0 Relay Control Register | Read/Write | base + $20_{16}$ |
| Bank 1 Relay Control Register | Read/Write | base + $22_{16}$ |
| Bank 2 Relay Control Register | Read/Write | base + $24_{16}$ |
| Bank 3 Relay Control Register | Read/Write | base + $26_{16}$ |
| Bank 4 Relay Control Register | Read/Write | base + $28_{16}$ |
| Bank 5 Relay Control Register | Read/Write | base + $2A_{16}$ |
| Bank 6 Relay Control Register | Read/Write | base + $2C_{16}$ |
| Bank 7 Relay Control Register | Read/Write | base + $2E_{16}$ |
| Channels 0990 - 0996 Relay Control | Read/Write | base + $30_{16}$ |

# Reading the Registers

Figures 1-1 and 1-2 (see Chapter 1) show the channels grouped by banks. You can read these Relay Matrix Switch registers:

- Manufacturer ID Register (base + $00_{16}$)
- Device Type Register (base + $02_{16}$)
- Status/Control Register (base + $04_{16}$)
- Bank 0 Relay Control Register (base + $20_{16}$)
- Bank 1 Relay Control Register (base + $22_{16}$)
- Bank 2 Relay Control Register (base + $24_{16}$)
- Bank 3 Relay Control Register (base + $26_{16}$)
- Bank 4 Relay Control Register (base + $28_{16}$)
- Bank 5 Relay Control Register (base + $2A_{16}$)
- Bank 6 Relay Control Register (base + $2C_{16}$)
- Bank 7 Relay Control Register (base + $2E_{16}$)
- Channels 0990 - 0996 Relay Control Register (base + $30_{16}$)

## Manufacturer Identification Register

The Manufacturer Identification Register is a read-only register at address $00_h$ (Most Significant Byte (MSB)) and $01_h$ (Least Significant Byte (LSB)). Reading this register returns the Hewlett-Packard identification, $FFFF_{16}$.

### Device Identification Register

The Device Identification Register is a read-only register accessed at address $02_{16}$. Reading this register returns module identification of 256 ($0100_{16}$) for an E1468A/E1469A module.

### Status/Control Register

The Status/Control Register informs the user about the module's status and configuration. Each relay requires about 12 msec execution time during which time the modules are "busy". Bit 7 of this register is used to inform the user of a "busy" condition. The interrupt generated after a channel has been closed can be disabled. Bit 6 of this register is used to inform the user of the interrupt status.

In addition, if a terminal module is connected to the switch module, the present configuration of the terminal module's status bit can be read. Bits 10, 11, 12, and 13 of this register are used to determine the configuration of the terminal module. For example, if the Relay Matrix Switch module is not busy (bit 7), the interrupt is enabled (bit 6), then a read of the Status/Control Register (base + $04_{16}$) returns DBBF.

### Relay Control Registers

Reading these registers always returns $FFFF_{16}$.

## Writing to the Registers

You can write to these Relay Matrix Switch module registers:

- Status/Control Register (base + $04_{16}$)
- Bank 0 Relay Control Register (base + $20_{16}$)
- Bank 1 Relay Control Register (base + $22_{16}$)
- Bank 2 Relay Control Register (base + $24_{16}$)
- Bank 3 Relay Control Register (base + $26_{16}$)
- Bank 4 Relay Control Register (base + $28_{16}$)
- Bank 5 Relay Control Register (base + $2A_{16}$)
- Bank 6 Relay Control Register (base + $2C_{16}$)
- Bank 7 Relay Control Register (base + $2E_{16}$)
- Channels 0990 - 0996 Relay Control Register (base + $30_{16}$)

### Status/Control Register

Writes to the Status/Control Register (base + $04_{16}$) enable you to disable/enable the interrupt generated when channels are closed. Writing a 1 to bit 0 of the Status/Control Register (base + $04_{16}$) does not change the state of the latching relays (individual channel relays). Writing a 1 to this bit has the same effect as removing power from the cardcage. Since the relays are latching relays, they do not change state.

**NOTE** *It is necessary to write a 0 to bit 0 after the reset has been performed before any other commands can be programmed and executed. SCPI commands take care of this automatically.*

To disable the interrupt generated when channels are closed, write a 1 to bit 6 of the Status/Control Register (base + $04_{16}$).

**NOTE** *Typically, interrupts are only disabled to "peek-poke" a module. Refer to the operating manual of the command module before disabling the interrupt.*

## Relay Control Registers

Writes to the Relay Control Registers (base + $20_{16}$ to base + $30_{16}$) enable you to switch desired channels. Figures 1-1 and 1-2 (see Chapter 1) show the schematics for the modules and the bank, row, and column information. Any number of relays per bank can be closed at a time.

**Manufacturer ID Register**

| base + $00_{16}$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Write | Undefined | | | | | | | | | | | | | | | |
| Read* | Manufacturer ID | | | | | | | | | | | | | | | |
| *Returns $FFFF_{16}$ = Hewlett-Packard A16 only register based. | | | | | | | | | | | | | | | | |

**Device Type Register**

| base + $02_{16}$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Write | Undefined | | | | | | | | | | | | | | | |
| Read | $0100_{16}$ | | | | | | | | | | | | | | | |

**Status/Control Register**

| base + $04_{16}$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Write* | Undefined | | | | | | | | | D | Undefined | | | | | R |
| Read** | Undefined | | S4 | S3 | S2 | S1 | Undefined | | B | D | Undefined | | | | | |

*R = Latching relays stay in their current state.
*D = Disable interrupt by writing 1 in bit #6.
**B = Status "busy" is 0 in bit #7.
**D = Status "Interrupt disable" is 1 in bit #6.
**S4 -S1 = Status "Configuration Status bits" hardwired onto the terminal modules.

| S4 | S3 | S2 | S1 | | |
|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | = | E1469A 4x16 Matrix |
| 0 | 1 | 0 | 1 | = | E1468A 8x8 Matrix |

**Bank 0 Relay Control Register**

| base + $20_{16}$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Write* | Undefined | | | | | | | | CH7 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 | CH0 |
| Read | Always Returns $FFFF_{16}$ | | | | | | | | | | | | | | | |

*Writes a 1 to close channel.

**Bank 1 Relay Control Register**

| base + $22_{16}$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Write* | Undefined | | | | | | | | CH7 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 | CH0 |
| Read | Always Returns $FFFF_{16}$ | | | | | | | | | | | | | | | |

*Writes a 1 to close channel.

**Bank 2 Relay Control Register**

| base + $24_{16}$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Write* | Undefined | | | | | | | | CH7 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 | CH0 |
| Read | Always Returns $FFFF_{16}$ | | | | | | | | | | | | | | | |

*Writes a 1 to close channel.

**Bank 3 Relay Control Register**

| base + $26_{16}$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Write* | Undefined | | | | | | | | CH7 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 | CH0 |
| Read | Always Returns $FFFF_{16}$ | | | | | | | | | | | | | | | |

*Writes a 1 to close channel.

**Bank 4 Relay Control Register**

| base + $28_{16}$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Write* | Undefined | | | | | | | | CH7 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 | CH0 |
| Read | Always Returns $FFFF_{16}$ | | | | | | | | | | | | | | | |

*Writes a 1 to close channel.

**Bank 6 Relay Control Register**

| base + $2C_{16}$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Write* | Undefined | | | | | | | | CH7 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 | CH0 |
| Read | Always Returns $FFFF_{16}$ | | | | | | | | | | | | | | | |

*Writes a 1 to close channel.

**Bank 7 Relay Control Register**

| base + $2E_{16}$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Write* | Undefined | | | | | | | | CH7 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 | CH0 |
| Read | Always Returns $FFFF_{16}$ | | | | | | | | | | | | | | | |

*Writes a 1 to close channel.

**Channels 0990 - 0996 Relay Control Register**

| base + $30_{16}$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Write* | Undefined | | | | | | | | | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 | CH0 |
| Read | Always Returns $FFFF_{16}$ | | | | | | | | | | | | | | | |

*Writes a 1 to close channel.

# Appendix C
# Relay Matrix Switch Error Messages

This table lists the error messages associated with the Relay Matrix Switch modules when programmed with SCPI. See the appropriate command module user's manual for complete information on error messages.

| Number | Title | Potential Cause(s) |
|---|---|---|
| -211 | Trigger ignored | Trigger received when scan not enabled. Trigger received after scan complete. Trigger too fast. |
| -213 | Init Ignored | Attempting to execute an INIT command when a scan is already in progress. |
| -224 | Illegal parameter value | Attempting to execute a command with a parameter not applicable to the command. |
| -350 | Too many errors. | The queue holds a maximum of 30 error numbers/messages for each switchbox. The queue has overflowed. |
| 1500 | External trigger source already allocated | Assigning an external trigger source to a switchbox when the trigger source has already been assigned to another switchbox. |
| 2000 | Invalid card number | Addressing a module (card) in a switchbox that is not part of the switchbox. |
| 2001 | Invalid channel number | Attempting to address a channel of a module in a switchbox that is not supported by the module (e.g.,, channel 99 of a multiplexer module). |
| 2006 | Command not supported on this card | Sending a command to a module (card) in a switchbox that is unsupported by the module. |
| 2008 | Scan list not initialized | Executing a scan without the INIT command. |
| 2009 | Too many channels in channel list | Attempting to address more channels than available in the switchbox. |
| 2012 | Invalid Channel Range | Invalid channel(s) specified in SCAN <*channel_list*> command. Attempting to begin scanning when no valid channel list is defined. |
| 2600 | Function not supported on this card | Sending a command to a module (card) in a switchbox that is not supported by the module or switchbox. |
| 2601 | Channel list required | Sending a command requiring a channel list without the channel list. |

## *Notes:*

# Appendix D
# Relay Life

## Replacement Strategy

Electromechanical relays are subject to normal wear-out. Relay life depends on several factors. The replacement strategy depends on the application. If some relays are used more often or at a higher load than other relays, the relays can be individually replaced as needed.

If all relays see similar loads and switching frequencies, the entire circuit board can be replaced when the end of relay life approaches. The sensitivity of the application should be weighed against the cost of replacing relays with some useful life remaining.

**NOTE** *Relays that wear out normally or fail due to misuse should not be considered defective and are not covered by the product's warranty.*

## Relay Life Factors

Some effects of loading and switching frequency on relay life follow.

- **Relay Load.** In general, higher power switching reduces relay life. In addition, capacitive/inductive loads and high inrush currents (for example, turning on a lamp or starting a motor) reduces relay life. Exceeding specified maximum inputs can cause catastrophic failure.
- **Switching Frequency.** Relay contacts heat up when switched. As the switching frequency increases, the contacts have less time to dissipate heat. The resulting increase in contact temperature also reduces relay life.

## End-of-Life Determination

A preventive maintenance routine can prevent problems caused by unexpected relay failure. The end of life of a relay can be determined by using one or more of three methods: contact resistance maximum value, contact resistance variance, and/or number of relay operations. The best method (or combination of methods), as well as the failure criteria, depends on the application in which the relay is used.

- **Contact Resistance Maximum Value.** As the relay begins to wear out, its contact resistance increases. When the resistance exceeds a predetermined value, the relay should be replaced.
- **Contact Resistance Variance.** The stability of the contact resistance decreases with age. Using this method, the contact resistance is measured several (5-10) times, and the variance of the measurements is determined. An increase in the variance indicates deteriorating performance.
- **Number of Relay Operations.** Relays can be replaced after a predetermined number of contact closures. However, this method requires knowledge of the applied load and life specifications for the applied load.

# Index
## E1468A/E1469A Relay Matrix Switch User's Manual


### A

ABORt subsystem, 40
addressing registers, 69
  ARM subsystem, 41
  ARM:COUNt, 41
  ARM:COUNt?, 42

### B

base address, register, 70

### C

cautions, 15
checking module identification, 33
command reference, 39
common commands
  *CLS, 65
  *ESE, 65
  *ESE?, 65
  *ESR?, 65
  *IDN?, 65
  *OPC, 65
  *OPC?, 65
  *RCL, 65
  *RST, 65
  *SAV, 65
  *SRE, 65
  *SRE?, 65
  *STB?, 65
  *TRG, 65
  *TST?, 65
  *WAI, 65
  format, 37
  quick reference, 65
configuring the switches, 15
connector pinouts, 11

### D

declaration of conformity, 9
detecting error conditions, 35
Device Identification register, 73
documentation history, 8

### E

error messages, 77
examples
  Advancing Scan Using TRIGger, 62
  Closing Switch Channels, 51

### E (continued)

examples (cont'd)
  Enabling a Single Scan, 44
  Enabling Continuous Scanning, 44
  Enabling ECL Trigger Bus Line 0, 46
  Enabling Operation Status Register Bit 8, 57
  Enabling Trig Out Port, 47
  Enabling TTL Trigger Bus Line 7, 48
  Identifying Relay Matrix Switch Modules, 33
  Illegal Channel Closure Error, 35
  initial operation, 29
  Opening Channels, 52
  Opening/Closing Rows/Columns, 34
  Querying Channel Closures, 51
  Querying Channel Open State, 53
  Querying Continuous Scanning State, 44
  Querying ECL Trigger Bus Enable State, 46
  Querying Number of Scans, 42
  Querying Operation Status Enable Register, 57
  Querying Trig Out Port Enable State, 47
  Querying Trigger Source, 64
  Querying TTL Trigger Bus Enable State, 49
  Querying the Error Queue, 61
  Reading Card #1 Model Number, 60
  Reading Card#1 Description, 59
  Reading the Operation Status Register, 58
  Saving and Recalling States, 35
  Scanning Channels, 54
  Scanning Using Bus Triggers, 64
  Scanning Using External Triggers, 64
  Sequencing Channels (E1468A), 34
  Sequencing Channels (E1469A), 34
  Setting Card #1 Module to Power-On State, 60
  Setting Ten Scanning Cycles, 41
  Stopping a Scan with ABORt, 40
  Synchronizing a Relay Matrix Switch, 36
  Using Interrupts to Signal Errors, 35

### I

initial operation, 29
INITiate subsystem
  INITiate:CONTinuous, 43
  INITiate:CONTinuous?, 44
  INITiate[:IMMediate], 44
installing switches, 18
interrupt priority, setting, 17

## L

linking commands, 39
logical address switch, setting, 16

## M

Manufacturer ID register, 72
matrixes,creating larger, 23

## O

OUTPut subsystem
- OUTPut:ECLTrg[:STATe], 45
- OUTPut:ECLTrg[:STATe]?, 46
- OUTPut[:EXTernal][:STATe], 46
- OUTPut[:EXTernal][:STATe]?, 47
- OUTPut:TTLTrg[:STATe], 48
- OUTPut:TTLTrg[:STATe]?, 49

## P

power-on/reset conditions, 32
programming
- addressing switches, 28
- register-based, 69
- using SCPI, 28

## Q

querying switches, 32

## R

recalling/saving states, 34
register-based programming, 69
registers
- addressing, 69
- base address, 70
- Device Identification, 73
- Manufacturer ID, 72
- Relay Control, 74
- Status/Control, 73

relay control registers, 74
relay life, 79
relay matrix switches
- addressing, 28
- checking module identification, 33
- commands, 31
- configuring, 15
- connector pinouts, 11
- description, 11
- detecting error conditions, 35
- error messages, 77
- initial operation, 29
- installing, 18
- power-on/reset conditions, 32
- programming, 28
- querying, 32
- specifications, 67
- switching channels, 33
- synchronizing, 36

relays
- end-of-life determination, 79
- life factors, 79
- replacement strategy, 79

restricted rights statement, 7
[ROUTe:] subsystem
[ROUTe:]CLOSe, 50
- [ROUTe:]CLOSe?, 51
- [ROUTe:]OPEN, 52
- [ROUTe:]OPEN?, 53
- [ROUTe:]SCAN, 53

## S

safety symbols, 8
SCPI commands
- command reference, 39
- quick reference, 66

SCPI, using, 28
setting logical address switch, 16
specifications, 67
Status register switch, setting, 16
status/control register, 73
STATus subsystem
- STATus:OPERation:CONDition?, 56
- STATus:OPERation:ENABle, 57
- STATus:OPERation:ENABle?, 57
- STATus:OPERation[:EVENt]?, 58
- STATus:PRESet, 58

switch descriptions, 11
switching channels, 33
synchronizing switches, 36
SYSTem subsystem
- SYSTem:CDEScription?, 59
- SYSTem:CPON, 59
- SYSTem:CTYPe?, 60
- SYSTem:ERRor?, 60

## T

terminal modules
- attaching to relay switch module, 27
- configuring/wiring, 20

TRIGger subsystem
- TRIGger:SOURce, 63
- TRIGger:SOURce?, 64
- TRIGger[:IMMediate], 62

## W

WARNINGS, 8, 15
warranty statement, 7